НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Прогнозування фродових транзакцій»

Виконав:
студент IV курсу, групи ІО-62
Гомонець Іван Іванович _____

Керівник:
Професор, д.т.н,
Новотарський Михайло Анатолійович _____

Консультант з н. контроль:
Професор, д.т.н,
Сімоненко Валерій Павлович _____

Рецензент:
к.т.н, доцент. кафедри СКС ФПМ
Орлова Марія Миколаївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Гомонцю Івану Івановичу

1. Тема проєкту «Прогнозування фродових транзакцій», керівник проєкту Новотарський Михайло Анатолійович, професор, д.т.н, затверджені наказом по університету від « 7 » травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 2020 р.

3. Вихідні дані до проєкту технічна документація і теоретичні дані

4. Зміст пояснювальної записки: опис предметної області і сучасних методів розв'язання поставленої задачі, опис та порівняння алгоритмів виявлення аномальних транзакцій, загальна структура системи. Оцінка точності моделі

5. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Н. контроль	Сімоненко В.П., професор, д.т.н.		

6. Дата видачі завдання 01.09 .2019 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	10.12.2019-14.12.2019	
2.	Вивчення та аналіз завдання	14.12.2019-15.03.2020	
3.	Розробка архітектури та загальної структури системи	15.03.2020-15.04.2020	
4.	Вибір алгоритмів для реалізаіції моделі	15.04.2020-30.04.2020	
5.	Програмна реалізація системи	30.04.2020-13.05.2020	
6.	Оформлення пояснювальної записки	13.05.2020-19.05.2020	
7.	Захист програмного продукту	20.05.2020	
8.	Передзахист	26.05.2020	
9.	Захист	25.06.2020	

Студент Іван ГОМОНЕЦЬ

Керівник Михайло НОВОТАРСЬКИЙ

АНОТАЦІЯ

В даній бакалаврській дипломній роботі спроектовано і реалізовано систему виявлення шахрайських транзакцій на основі історичних даних із застосуванням моделі машинного навчання.

Система може бути інтегрована в сервіси онлайн-оплати і блокувати транзакції, ініційовані шахраями.

Програмна частина реалізована скриптовою мовою програмування Python із використанням спеціальних бібліотек для обробки даних.

ANNOTATION

In the given bachelor thesis work, a system for detecting fraudulent transactions based on historical data using a machine learning model is designed and implemented.

The system can be integrated into online payment services and block transactions initiated by fraudsters.

The software part is implemented in the Python scripting language using special libraries for data processing.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.467100.001 ВП	Відомість проекту	1	
3	А4	ІАЛЦ.467100.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.467100.003 ПЗ	Пояснювальна записка	65	
5	А4	ІАЛЦ.467100.004 Д1	Діаграма класів	1	
6	А4	ІАЛЦ.467100.005 Д2	Структурна схема системи	1	
7	А4	ІАЛЦ.467100.006 Д3	Функціональна схема алгоритму моделі	1	
8	А4	ІАЛЦ.467100.007 Д4	Текст програми	5	

					ІАЛЦ.467100.001 ВП		
Зм.		*№ документа*	*Підп.*	*Дата*			
Розробив		*Гомонець І.І.*			Відомість дипломного проєкту	Літ.	*Аркуш* 1 / *Аркушів* 1
Перевірив		*Новотарський М.А.*				Т	
Н.контр.		*Сімоненко В. П.*				*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ гр. ІО-62*	
Затв.		*Стіренко С. Г.*					

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи

освітньо-кваліфікаційного рівня бакалавр

на тему: « Прогнозування фродових транзакцій»

Київ – 2020 рок

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку курсу «Інженерія програмного забезпечення» та курсу «Захист інформації у комп’ютерних системах». Область застосування: практичне використання у бізнесі, що надає послуги онлайн оплати.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи прогнозування шахрайських транзакцій на етапі процесингу.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії і практики програмування, обробки даних, наукові статті, публікації в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробляємого продукту

- Система може оброблювати дані лише для того клієнта, звідки взяті історичні дані.

5.2. Вимоги до програмного забезпечення

- Наявність на сервері Python не нижче версії 3.6.
- Наявність на сервері Flask не нижче версії 1.1.2.
- Наявність на сервері scikit-learn не нижче версії 20.0.

5.3. Вимоги до апаратної частини.

- Сервер із підтримкою розгортання додатків на Python
- Оперативної пам’яті не менше 512 Мбайт.
- Стабільне підключення до мережі Інтернет.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	30.03.2020
Складання і узгодження технічного завдання	01.04.2020
Створення структури системи і окремих модулів	10.04.2020
Тестування окремих модулів системи	21.04.2020
Допрацювання, налагодження і виправлення помилок	06.05.2020
Оформлення документації дипломної роботи	20.05.2020

Пояснювальна записка
до дипломного проєкту
на тему: «Прогнозування фродових транзакцій»

Київ - 2020 року

ЗМІСТ

ВСТУП

На сьогодні актуальним є проведення оплат послуг та товарів в мережі Інтернет. Все більше користувачів використовують сервіси інтернет оплати для розрахунку в магазинах, іграх, додатках та веб-порталах. Відповідно, такий тренд посилює навантаження на систему та ставить нові вимоги до забезпечення безпеки проведення транзакцій онлайн та відслідковування і прогнозування таких транзакцій з метою запобігання втрати коштів клієнтів.

Актуальність теми. Через зростання популярності онлайн-оплати (дебетові карти, онлайн-гаманці, криптовалюти) у 2020 році на 18% проти 2019 року зростають і втрати від шахрайства. Такий відчутний ріст спричинений збільшенням сервісів з оплатою оналайн та, особливо, можливістю смартфонів, які ні в чому вже не поступаються персональним комп'ютерам при здійсенні транзакцій. Також активно відбувається процес діджиталізації країн Африки, що призведе до ще більшого росту інтернет-користувачів, а отже і кількості оплат в мережі.

Мета і задачі дослідження. Метою даної роботи є розробка моделі, основаної на машинному навчані для виявлення і прогнозування шахрайських транзакцій на етапі здійснення оплати з подальшим її блокуванням.

Для виконання заданої мети були поставлені наступні основні задачі:

- Провести огляд вже готових систем для виявлення аномальних транзакцій, вияснити чи надають вони змогу передбачити такі транзакції до моменту оплати банком;
- Дослідити дані і навчити модель на історичних даних;
- Провести тестове моделювання розробленої моделі;
- Дослідити і зробити висновки по отриманих результатах моделювання на тестових даних, особливо по повноті прогнозу саме шахрайських транзакцій.

Практичне значення. Розроблена модель дозволить ефективно прогнозувати шахрайські транзакції ще до моменту їх передачі на

опрацювання банками. Також із застосуванням машинного начання вдасться виявляти транзакції, які не можуть бути помічені як фродові звичайними евристичними способами. Це дозволить компаніям звільнити людські ресурси для аналізу більш вузьких місць на проекті, зменшити вплив людського фактору та мінімізувати втрати.

У розділі 1 проведенно огляд існуючих рішень щодо виявлення шахрайських транзакцій. Наведено порівняльну характерисику цих методів і систем, визнчено переваги та недоліки.

У розділі 2 визначено методи та їх математичне забезпечення, які були обрані для вирішення поставленної задачі. Розглянуто методи передобробки даних, знаходження зв’язків а також метрики оцінки точності моделі.

У розділі 3 описано та реалізовано структуру системи із застосуванням методів, які були визначені в розділі 2. Проведено оцінку моделі по метриках точності прогнозування. Наведено можливі способи вдосконалення і розширення системи.

Ключові слова. Модель, аналіз, дерево-рішень, обробка даних, робота з великими даними, масштабованість, точність, машинне навчання, кореляція.

РОЗДІЛ 1
ОГЛЯД ВЖЕ ГОТОВИХ СИСТЕМ ДЛЯ ВИЯВЛЕННЯ АНОМАЛЬНИХ ТРАНЗАКЦІЙ

Система AFS (Advanced Fraud Detection System) - це комплексне рішення, що забезпечує захист додатків Інтернет-банкінгу та їх користувачів від кіберзлочинності, виявлення шахрайства в онлайн-середовищі та реєстрації діяльності користувачів. AFS виявляє активність фінансових зловмисних програм, спроби придбання облікових записів, фішинг, шахрайські операції, несанкціонований доступ до функцій, незахищені конфігурації та мережі користувачів, спроби атак веб-додатків, спроби автоматизації, тощо [1].

1.1. Система ThreatMark

На момент написання роботи однією з кращих систем для виявлення фродових транзакцій вважається ThreatMark.

Традиційний підхід до реалізації рішень AFS – система складається з багатьох різних модулів (аналіз поведінки користувача, операції, аналіз мережі, аналіз підключення, протидія зняття коштів та інші), які збираються в єдине ціле силами розробників. Очевидною перевагою таких систем є швидкий обмін даними, управління процесами, персоналізованими налаштуваннями в порівнянні з ThreatMark. Основним недоліком таких систем є складна реалізія, а також потрібно більше часу для інтеграції та налаштування.

Традиційні рішення також не можуть протидіяти сучасним загрозам, таким як атака з нулевим днем, соціальна інженерія, фішинг, атака із застосуванням персонального браузера.

Застосування традиційних рішень також відкидає збирання біометричних показників користувача, так званих його відбитків пальців в мережі – звідси випливає проблема частоти їх виявлення та збільшення

помилок другого роду, помилковий позитивний результат. Певні системи все ж аналізують поведінку користувача в мережі, але роблять це досить примітивними способами, а тому зловмисники можуть легко їх обійти такими способами:

- Зловмисники можуть перед атакою використовувати свої пристрої і виконувати дії з мінімальним рівнем ризику. Нарешті, коли буде проведена справжня шахрайська транзакція, система її не розпізнає, оскільки вона буде зроблена з довіреного пристрою.
- Може бути використане троянське програмне забезпечення за допомогою якого шахрай отримує контроль над пристроєм жерви і здійснює типові для неї транзакції.
- Використання автоматичних сценаріїв, які дозволяють у фоновому режимі під час сеансу жерви проводити задані зловмисником транзакції.

ThreatMark AFS був розроблений як альтернатива традиційним аналітичним системам, які не є адаптованими до швидких змін і нових загроз, які з’являються щодня. Також можна відмітити високу швидкодію даної системи. На рис. 1.1 – скрін інтерфейсу адміністратора ThreatMark [2].

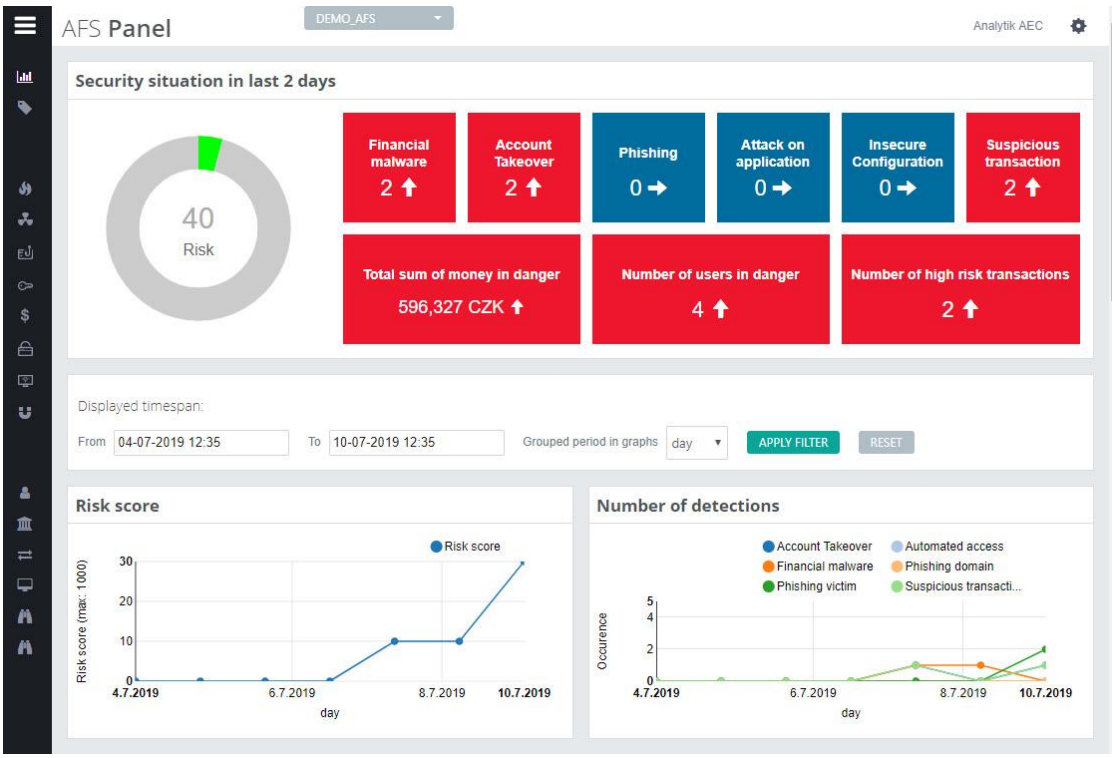


Рис. 1.1. Скрін інтерфейсу адміністратора ThreatMark [2]

Однією з корисних функції системи ThreatMark є ідентифікація і деталізація активностей на рівні користувача. Дашборд особливо інформативний у ситуації, вирішення конфліктів «дружнього фроду», коли клієнт хоче повернути гроші за використану послугу. На рис. 1.2 приведено демо-зріз такої деталізації [2].

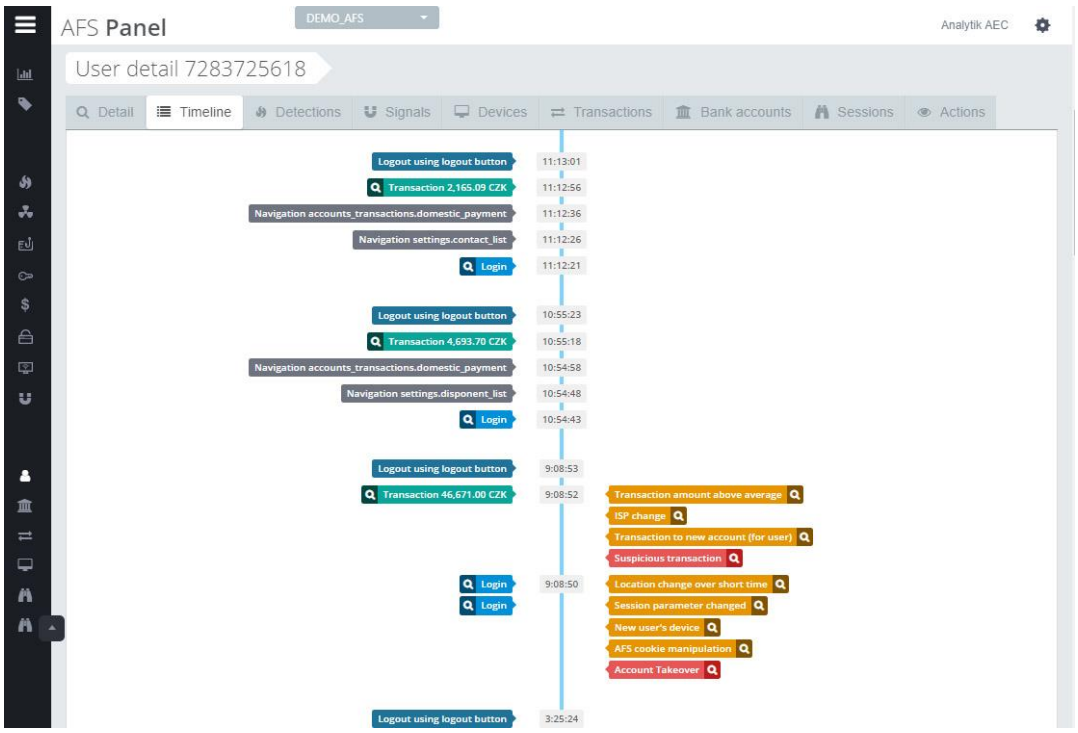


Рис.1.2. Деталізація активностей клієнта [2]

Для вибраної системи визначимо основні переваги і недоліки. До переваг можна віднести:

1. Рішення AFS розповсюджуэться як послуга. При цьому воно потребує для роботи широкий стек технологій, які працюють на операційній системі Linux. Для зібрання окремої версії AFS необхідно визначити її код, конкретні версії бібліотек, опен-сорс програм та конфігураційних файлів мережі. Отже, система AFS не може бути запущена в роботу на випадковому сервері.
2. «Підхід чорної скриньки» до системи дозволяє швидко та детерміновано вносити технічні і структурні зміни в систему одночасно у всіх підключених клієнтів. Дана опереція можлива завдяки простому оновленю системи на стороні клієнта до нової версії. Звідси компанія отримує набагато більше вільного часу на розробку новітніших версій та вдосконалень, а не займається оновленням ПЗ для кожного окремого клієнта. Даний підхід дозволяє зменшити вартість обслуговування, оскільки система легко розповсюджується.

3. Як було вже вказано, нова версія продукту оновлюється разом у всіх клієнтів. Наприклад, FDS оновлюється один раз в рік, а саме відбувається зміна програмних компонентів на декількох серверах одночасно. У некерованому ручному режимі, цей процес потребує зусиль спеціалістів на стороні клієнта (наприклад, банку)[2].

До недоліків розглянутої системи можна віднести:

1. Завищена ціна користування сервісом (0.01 долара за одну здійснену транзакцію)
2. Сценарії, які включає система, обмежені

1.2. Характеристика підсистем Clair та Anti-Fraud Suite

У лінійці продуктів ThreatMark є дві підсистеми для виявлення аномальних транзакцій:

1. Clair
2. Anti-Fraud Suite

ThreatMark Clair – платформа SaaS (сервіс як бізнес) використовується для прогнозування та зменшення кількості шахрайств при онлайн кредитуванні клієнтів. Система використовує так звані біометричні та поведінкові фактори користувача в мережі і на їх основі дозволяє визначати злочинців в режимі реального часу. [3].На рис. 1.3 зображено інтерфейс підсистеми Clair.

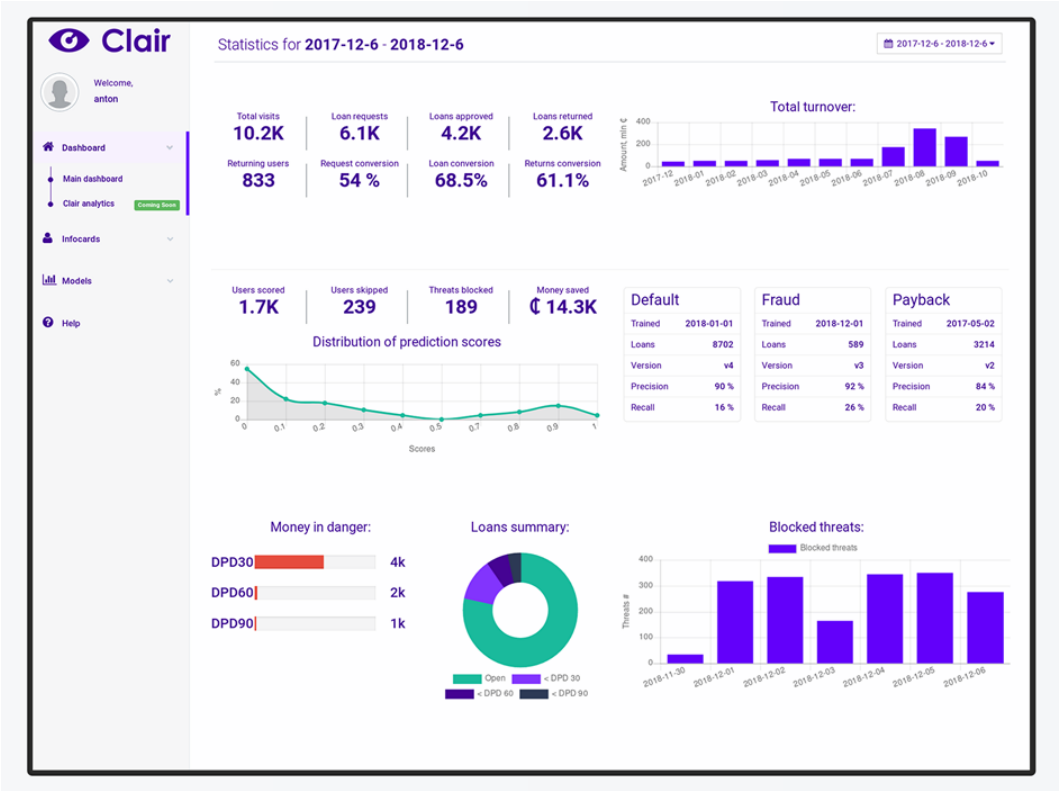


Рис.1.3. Підсистема Clair [3]

Основні переваги і недоліки даної підсистеми. Перевагами Clair можна виділити:

1. Прогнозування в режимі реального часу.
2. Збирання і агрегація даних на стороні сервісу, використовуючи JavaScript.
3. Висока точність прогнозу, мінімальна кількість помилково позитивних результатів.
4. Легкий і зрозумілий інтерфейс для управління. Не потрібно спеціально навчених людей.

До недоліків можна віднести:

1. Висока вартість обробки однієї транзакції
2. Складність інтеграції у вже існуючу систему оплати

Anti-Fraud Suite - рішення щодо запобігання шахрайству наступного покоління ThreatMark було створено як відповідь на болі в галузі фінансових послуг, постійно стикаючись з різними формами кіберзлочинності, шахрайства та регуляторних проблем. Ми виявили, що існуючі системи виявлення шахрайства не можуть конкурувати з постійно мінливим ландшафтом шахрайства та новими методами.

ThreatMark Anti-Fraud Suite ґрунтується на глибокому поведінковому профілюванні та машинному навчанні та забезпечує 360 ° цифровий банківський захист у всіх онлайн та мобільних каналах [4].

Основні переваги і недоліки підсистеми Anti-Fraud Suite. До переваг можна віднести:

1. Система працює на будь-якому пристрої та каналі через один механізм аналітики.
2. Налаштування відбувається в кожному випадку окремо, в залежності від вимог і потреб бізнесу.
3. Система модульна, отже її можна використовувати не тільки як цілий сервіс, але і окремі її частини для зміцнення і більшого захисту вже працюючої системи на проекті.

Серед недоліків можна виділити:

1. Дороговизна користування
2. Незважаючи на високу точність прогнозу помилок другого роду, загальна точність системи на рівні 70%

1.3. Огляд сучасних підходів обробки даних та визначення відхилень

Аномалії - це зразки даних, які не відповідають чітко визначеному поняттю нормальної поведінки. Рис. 1.4 ілюструє аномалії у простому двовимірному наборі даних. Дані мають дві нормальні області, N1 та N2, оскільки більшість спостережень лежать у цих двох областях. Точки, які досить віддалені від регіонів, наприклад, точки O1 та O2, і точки в області O3, є аномаліями [5].

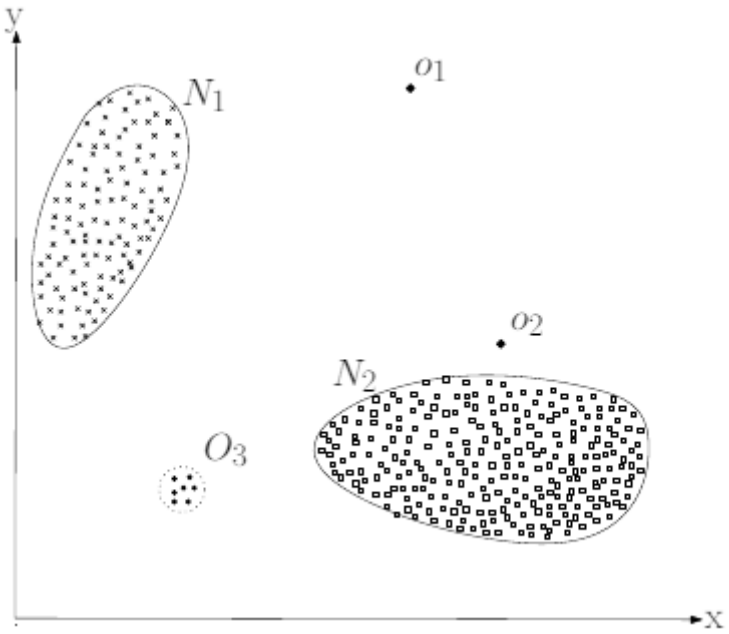


Рис.1.4. Простий приклад аномалій у 2-вимірному датасеті [5]

Виявлення аномалії є важливою проблемою, яка досліджувалася в різних областях науки та областях застосування. Багато методів виявлення відхилень були спеціально створені для певних областей застосувань, інші - більш загальні у застосуванні.

Визначення відхилення стосується проблеми знаходження шаблонів у даних, які не відповідають типовій поведінці. Ці невідповідні зразки часто називають аномаліями, відхиленнями, викидами, розбіжними спостереженнями, винятками в різних областях застосування.

З них, аномалії та відхилення - це два терміни, які найбільш часто застосовуються в контексті визначення аномалій.

Знаходження відхилень має широке використання у таких сферах як шахрайства за кредитними картками, страхування, охорона здоров'я, виявлення вторгнень для кібербезпеки та виявлення несправностей у критичних системах [5].

На даний момент можна виділити такі способи виявлення відхилень:

1. Ймовірнісний
   a. Параметричний
   b. Непараметричний
2. Способи, які спираються на класифікацію

1.3.1 Знаходження відхилень, використовуючи розподіл Гаусса

Гауссовим називають розподіл ймовірностей неперервної випадкової величини X, щільність якої має наступний вигляд:

$$f(x) = \frac{1}{\delta\sqrt{2\pi}} * e^{-\frac{(x-\mu)^2}{\delta^2}}$$

де:

μ − математичне сподівання,

σ − середнє квадратичне відхилення.

А ймовірність того, що X прийме значення, котре належить інтервалу (α, β):

$$P(\alpha < x < \beta) = \Phi\left(\frac{\beta - \mu}{\delta}\right) - \Phi(\frac{\alpha - \mu}{\delta})$$

де:

$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x * e^{-\frac{(-x)^2}{2}} dx$- функція Лапласа

Стандартний нормальний гауссовий розподіл – це дзвіноподібна крива розподілу ймовірності зі середнім значенням $\mu = 0$ та стандартним відхиленням $\sigma = 1$.

Відмітимо, що щільність розподілу вища за середнє значення і швидко набуває меншого значення в залежності відстані від середнього.

Тоді нехай заданий набір даних $\{x^{(1)}, x^{(2)} \ldots x^{(m)}\}$, визначимо наближено Гауссіський розподіл, застосувавши наступну оцінку параметра:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\delta^2 = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu)^2$$

Оцінка параметрів віддбувається за допомогою максимальної правдоподібності оцінок. Відстань конкретної точки в даних до оціночної середньої величини − це наближена оцінка відхилення для цієї точки в даних (під точкою розуміємо узагальнення від конкретних областей застосування). До оцінок відхилень використовується поріг для визначення відхилень в даних. Різні підходи в цій категорії по-різному знаходять відстань до середнього та порогу [6].

Тоді задамо правило, яке означатиме, чи є елементи нашої вибірки викидами:

$f(x_{tst}) < \varepsilon$ − умова за якої транзакція вважається викидом

$f(x_{tst}) \geq \varepsilon$ − умова за якої транзакція вважається нормальною

де:

ε − це будь-яке достатньо мале число.

$f(x_{tst})$ − ймовірність x_{tst} в гауссовому розподілі.

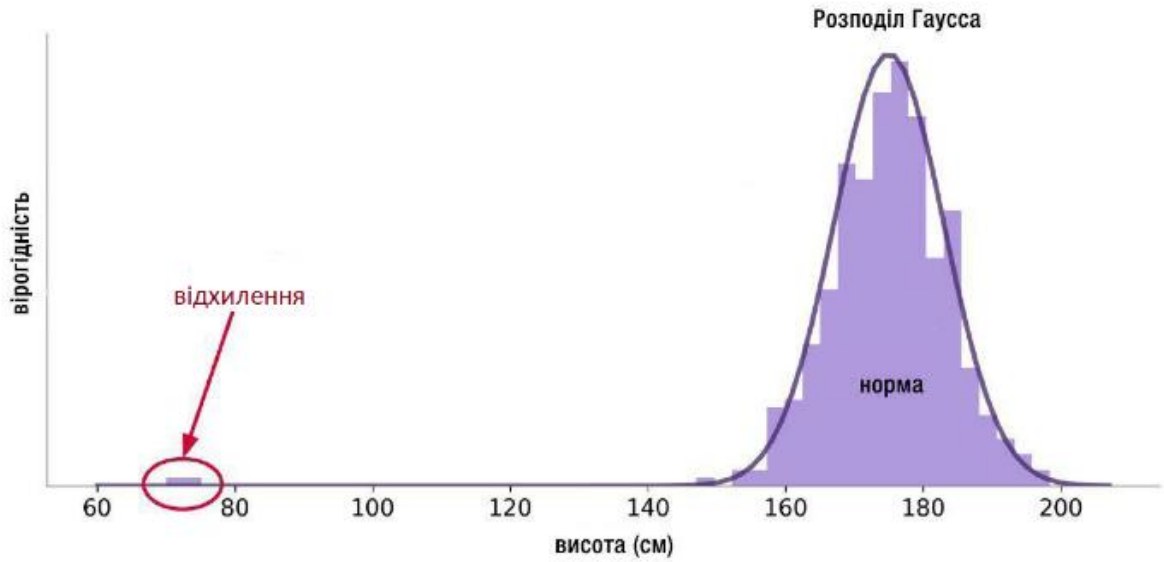


Рис.1.5. Приклад виявлення відхилення на основі нормального розподілу [7]

За відхилення ми приймаємо транзакцію, ймовірність якої приймає достатньо мале значення.

На основі згаданого методу, можна скласти алгоритм визначення такого відхилення:

1. На вході маємо набір даних, у якому знаходяться нормальні та шахрайські транзакції.
2. Розбиваємо дані на кілька вибірок − кожна з яких містить попарно два параметри: час і сума, час і місце розташування користувача, час і девайс користувача.
3. Обчислюємо ймовірності для кожного сформованоо набору даних
4. Будуємо графіки ймовірностей та визначаємо, яка з транзакцій була шахрайською [7].

1.3.2 Знаходження відхилень, використовуючи логістичну регресію

Логістична регресія – це підвид множинної регресії, загальне застосування якої полягає в аналізі і знаходженні зв'язку між кількома незалежними змінними і залежною цільовою змінною. Логістична регресія застосовується для розв’язання задач класифікації. Бінарна логістична регресія, використовується, якщо залежна цільова змінна є бінарною (тобто

може приймати на виході один або нуль). Конкретизуючи, з використанням логістичної регресії можна знаходити ймовірність того, що певна подія відбудеться для кожного конкретного зразка даних.

Регресійні моделі можна подати у вигляді простої формули:

$$y = F(x_1, x_2 \dots, x_n)$$

В множинній лінійній регресії прогнозується, що залежна цільова змінна є лінійною функцією її незалежних змінних і може бути обчислена наступним чином:

$$y = a_0 + \sum_{i=1}^{n} a_i x_i$$

де:

$a_0 \dots a_i \dots a_n$ – коефіцієнти моделі

x_i – вхідні фактори моделі

Слід відмітити, що однією з проблем є те, що при використанні данного способу немає можливості обчислити плавність змін, оскільки він є бінарним і приймає на виході значення нуль або один. Для вирішення цієї проблеми варто застосувати формулу логіт-перетворення:

$$P = \frac{1}{1 + e^{-y}},$$

де:

P – ймовірність того, що настане подія, яка нас цікавить у прогнозі

y – стандартне рівняння регресії

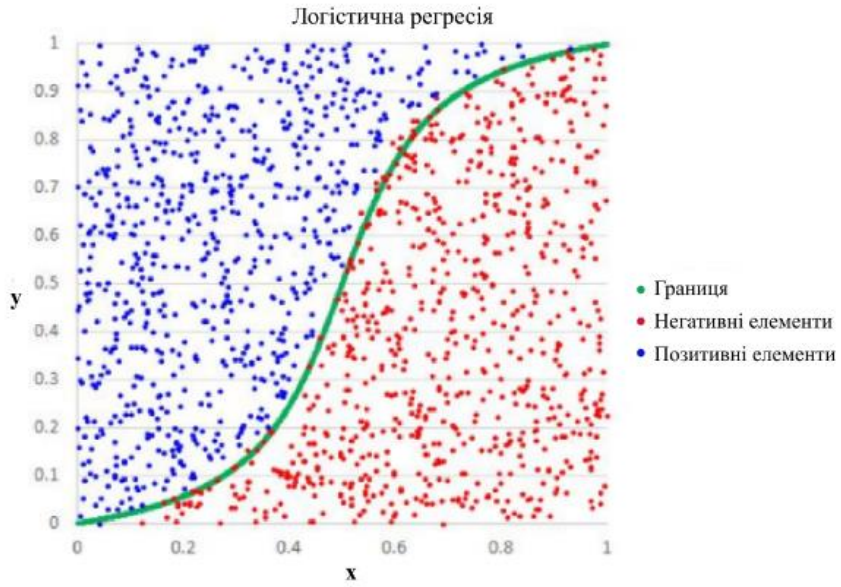


Рис.1.6. Застосування логістичної регресії [8]

В результаті застосування логіт-перетворення отримуємо результат, котрий знаходиться на проміжку від нуля до одиниці.

Основним недоліком застосування логістичної регресії є інтерпретація та фізичний зміст ваг, оскільки вони є мультиплікативними та не адитивними.

При повністю розподілених даних в навчальному датасеті є проблеми з навчанням моделі. Якщо є параметр, який відмінно поділяє цільову зміну на два класи, то начання моделі вже буде нереально зробити. Пояснюється це тим, що сума ваг цієї ознаки не збіжиться, тому що оптимальна вага буде величиною нескінченною. Проте, якщо в нас є ознака, яка розподіляє вибірку точно на два класи, то нам не потрібно застосовувати машинне навчання для обчислення такої задачі. Це нерозумно по використанню ресурсів та часу витраченого на розробку моделі. Проблему ідеального розподілу можна вирішити, ввівши пенальтизацію ваг, або визначивши попередні розподіли ймовірностей ваг.

До плюсів використання моделі логістичної регресії можна віднести те, що ми отримуємо не просто класифікацію об’єктів, але і ймовірності кожного класу. Це значна перевага над моделями, що повертають нам лише кінцевий остаточний результат у вигляді одного класу. Розуміння того, що ми маємо

ймовірність 99% відношення до класу проти 51% дуже важливе і може зіграти вирішальну роль у певних задачах.

Також модель логістичної регресії можна розширити і оптимізувати для вирішення задач багатокласової класифікації, а не просто бінарної [8].

1.3.3 Знаходження відхилень, використовуючи дерево розв'язання

Дерево розв'язання − це метод відображення послідовностей рішень в ієрархічній структурі, для кожному об'єкту якого є відповідний єдинний стан, що дає рішення.

Випадки, коли можна використовувати дерево прийняття рішень переважно мають наступний опис: є вибірка випадків, кожна з яких характеризується певним набором дискретних значень, і в кожному зразку ми знаємо значення деякої (невідомої) булевої функції, яка є залежною від цих значень. Тобто, ми повинні розробити певну логічну конструкцію, яка буде описувати задану функцію і дозволить класифікувати нові нерозмічені дані.

Слід відмітити, що для розробки дерева рішення потрібно знайти і обчислити умови, які розіб'ють вибірку на підмножини найоптимальнішим методом. Тобто, після розділення має утворитись дві підмножини, які будуть містити елементи, що відносяться до однакового класу, або щоб більшість із цих елементів належала до одного класу. [9]

Для формалізації визначимо поняття ентропії. Отже, нехай задана певна вибірка A, яка містить n елементів, m з них мають властивість S. Тоді ентропію вибірки A по відношенню до властивості S обчислимо із застосуванням наступної формули:

$$H(A,S) = -\sum_{i=1}^{s} \frac{m_i}{n} log \frac{m_i}{n}$$

Наступним важливим пунктом для дерева рішення є теоретико-інформаційний критерій, що відповідає за вибір відповідного атрибуту. Нехай задана певна вибірка елементів A, тоді припустимо,що у певних з цих значень існує властивість S, яка може бути класифікована з використанням атрибута

Q, який має q різних можливих значень. Отже, акумулятивна сума інформації обчислюється за такою формулою:

$$Gain(A, Q) = H(A, S) - \sum_{i=1}^{q} \frac{|A_i|}{A} H(A_i, S)$$

де A_i − це вибірка елементів A, при яких атрибут Q набуває значення i.

На кожному етапі, вибирається атрибут за допомогою алгоритму, щоб акумулятивна сума інформації мала якомога більше значення.

Ще один із найбільш важливих критеріїв для дерева рішення є індекс Джині. Критерій заснований на принципі, що для множини тестів A і властивості S, що набуває значень s, індекс обчислюється за формулою:

$$Gini(A, S) = 1 - \sum_{i=1}^{s} \frac{|A_i|}{A}$$

Відповідно, для множини тестів A і атрибуту Q, що набуває значень q і цільового значення S, що має s різних значень, індекс обчислюється за формулою:

$$Gini(A, Q, S) = Gini(A, S) - \sum_{i=1}^{q} \frac{|A_i|}{A} \cdot Gini(A, S)$$

Індекс Джині частково компенсує ухилення критерію приросту інформації для вибору більш розгалужених атрибутів. По інших критеріях ці два параметри дуже схожі між собою і розбіжність між ними становить не більше 2%. [10]

Приклад застосування моделі дерева рішення наведений на рисунку 1.7

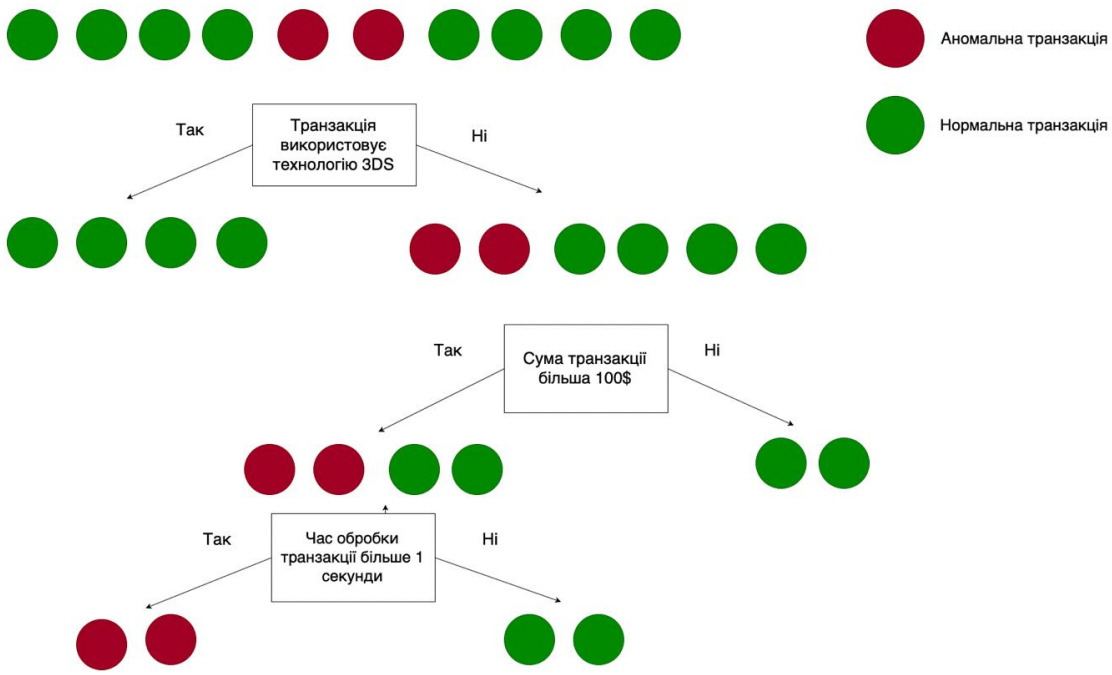


Рис.1.7. Приклад застосування дерева рішення

Проаналізувавши отримані дані створимо відповідний алгоритм:

1. На вході маємо датасет, якому знаходяться нормальні та шахрайські транзакції.
2. Використовуючи випадковий вибір, розділяємо вхідні дані на дві вибірки – тренувальну та тестову.
3. Визначаємо та вибираємо множину атрибутів, які послідовно виначатимуть, чи є транзакція шахрайською.
4. Обираємо треш-холди по яких встановлюємо кінець навчання дерева.

1.3.4 Знаходження відхилень, використовуючи випадковий ліс

Дерева рішень на час написання роботи є одними з кращих і універсальних методів класифікації для ігрової індустрії, тому що вони можуть досягати стовідсоткової точності на будь-якому датасеті.

Метод випадкових дерев рішень дозволяє мінімізувати кореляцію між цими деревами і уникнути перенавчання. Базові дерева навчаються на різних підвибірках ознак, які також вибираються випадковим чином.

Ансамбль моделей на основі дерев, що використовують випадкові підпростори, можна реалізувати за наступним алгоритмом:

1. Припустимо, що кількість овибірок для навчання – N, а кількість ознак – D.
2. Введемо L – число моделей в ансамблі.
3. Для кожної вибраної моделі *l* визначаємо *dl (dl < D)*, яке буде кількістю ознак для *l*. Для всіх визначених моделей застосовуємо одне значення *dl*
4. Для кожної вибраної моделі *l* формуємо випадковим чином тренувальну вибірку, обравши *dl* факторів з D та навчаємо модель на цих значеннях.
5. Для використання ансаблю моделей до нового об’єкту даних потрібно обрати мажоритарним мотодом клас, або усередненням значень для задач регресії.

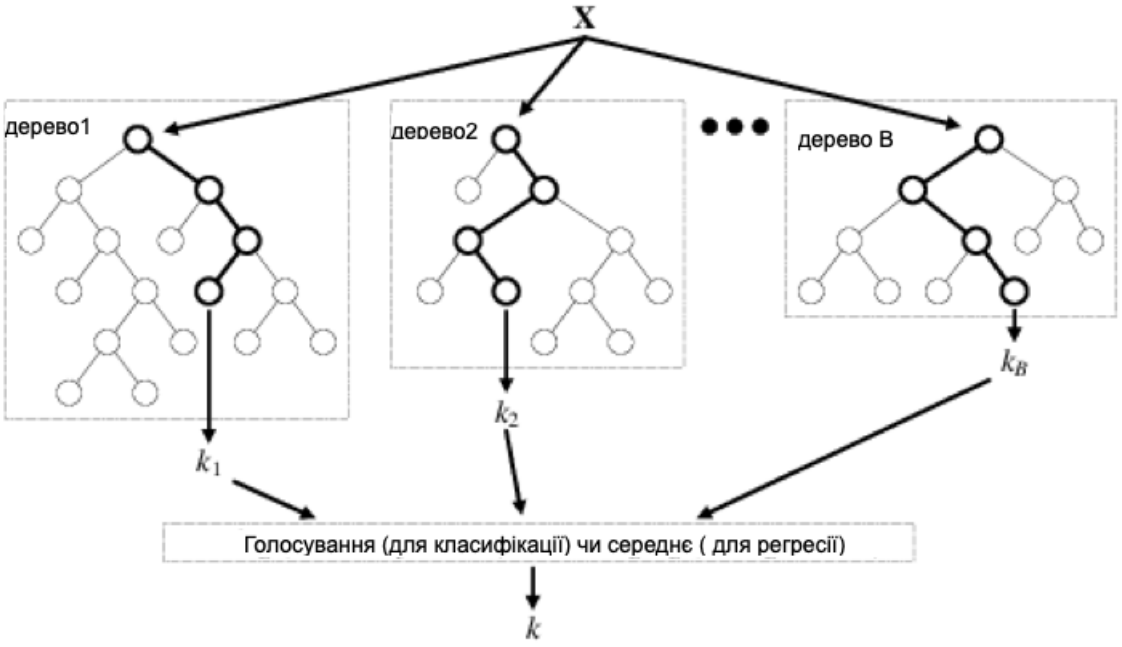


Рис.1.8. Приклад використання моделі випадкового лісу [11]

Для використання моделі випадкового лісу, яка будується на основі N дерев, застосуємо наступний алгоритм:

Для кожного n = 1,…, N:

1. Випадковим чином вибірку X_n за використанням бутстрапа
2. Будуємо дерево рішення b_n на основі множини X_n
    a. За вказаними критеріями обираємо найкращу ознаку та проводимо розділення — і так, поки не закінчиться вибірка.
    b. Будуємо дерева, поки не буде досягнуто умови n_{min} об’єктів, або необхідної глибини дерева.
    c. При кожному розбитті випадковим чином обирається m ознак з n вихідних, а найкраще розділення вибірки знаходить тільки одну.

Отриманий класифікатор:

$$a(x) = \frac{1}{N}\sum_{i=1}^{N} b_i(x)$$

Варто відмітити, що для задач класифікації випадковим чином обираємо $m = \sqrt{n}$, де n – це загальна кількість ознак в датасеті. Дерево рішення будується до моменту, коли на кожному листі буде один об’єкт [12, 13].

1.3.5 Знаходження відхилень, використовуючи K найближчих сусідів (KNN)

Модель k-найближчих сусідів (k Nearest Neighbor) — також один із популярних методів розв’язання задач класифікації об’єктів.

Припустимо, що ми на вхід отримали певний датасет, який містить об’єкти різних класів, але вони вже розподілені по цих класах. Основне завдання полягає в тому, що потрібно знайти евристику, яка дозволить нам для нового нерозміченого об’єкту даних визначити клас з уже відомих раніше.

Застосовуючи модель KNN, ми відносимо новий невідомий об’єкт до того класу, з яким досліджуваний об’єкт має найбільше сусідів. Під «сусідом» розуміємо інші об’єкти вибірки, які знаходяться найближче до досліджуваного об’єкту за певними ознаками. (Рис.1.9)

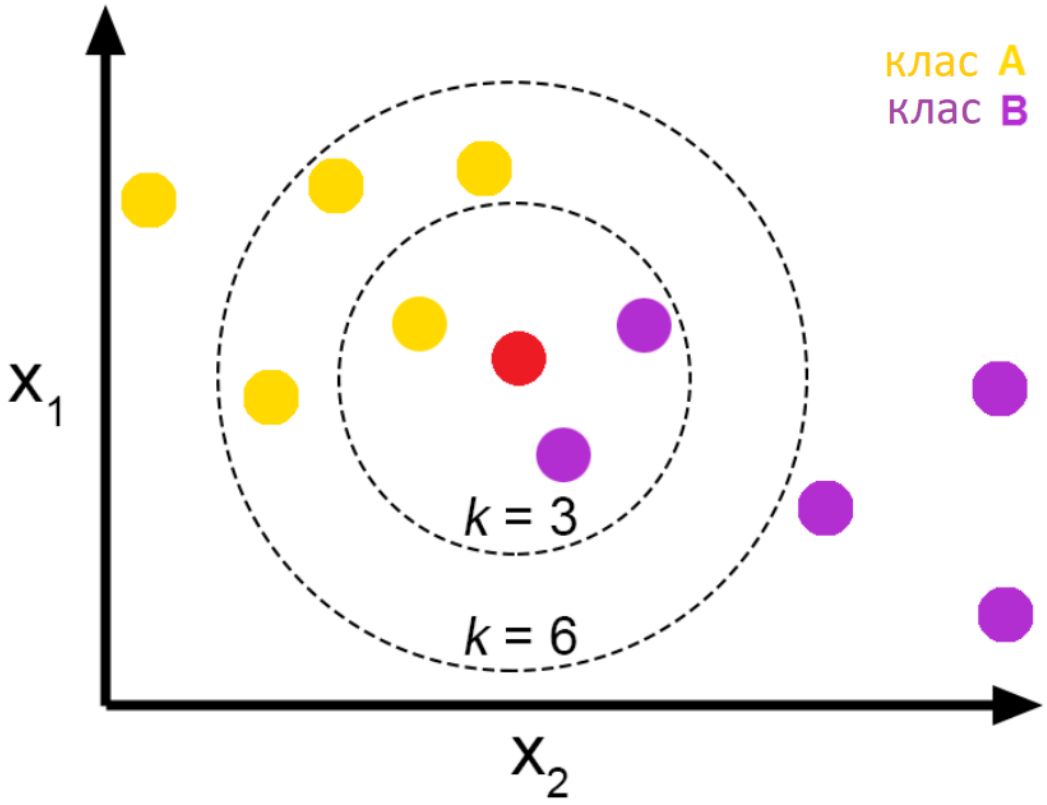


Рис.1.9. Приклад роботи моделі KNN

Для розрахунку відстані між об'єктами можуть бути застовані різні методи. Одним з них є застосування формули Евклідової відстані, яка може бути обчислена за допомогою такої формули:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} =$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

До плюсів класифікатора можна віднести:

1. Відносно простий алгоритм для навчання і інтерпретації результатів.
2. Може бути застований для розв'язання задач як класифікації, так і регресії.
3. Достатньо висока точність, але при таких же витратах ресурсів, є кращі моделі з контрольваним навчанням.
4. Корисний для роботи з нелінійними даними.

До мінусів можна віднести:

1. Ресурсновитратний алгоритм, оскільки при навчані зберігаються всі дані тренувальної вибірки.
2. З пунктуу один випливає вимога до високого об’єму пам’яті.
3. За умови досить великого N повільна обробка і прогнозування
4. Точність класифікації чутлива до масштабу в даних.

1.3.6 Порівняння розглянутих способів виявлення аномальних транзакцій

Порівняння по основних показниках розглянутих способів наведено в Таблиці 1.

Таблиця 1. Порівняння основних показників

Спосіб	Чутли-вість до вики-дів	Робота на невеликій тренуваль-ній вибірці	Масштабова-ність	Мультикласовість	Точність
Розподіл Гаусса	+	+	-	-	80-90%
Логістична регресія	-	-	+	+	84-90%
KNN	+	-	-	+	85-95%
Дерево рішення	+	-	+	+	80-90%
Випадковий ліс	+	-	+	+	85-95%
ThreatMark	+	-	+	-	75%
Clair	+	-	-	-	75%
Anti-Fraud Suite	+	-	-	-	70%

З таблиці видно, що випадковий ліс, k найближчих сусідів мають найбільш високу точність прогнозування. Варто також відмітити, що застосування розподілу Гаусса доцільне лише для невеликих наборів даних, які в свою чергу мають мати нормальний розподіл. Програмні методи не дають такої високої точності, але це вже готові рішення, які можуть бути інтегровані менше , ніж за два тижні. Також вони розроблені тільки для конкретних цілей, а отже для сегмениації клієнтів наприклад, вже не можуть бути використані.

ВИСНОВКИ ДО РОЗДІЛУ 1

В першому розділі були оглянуті існуючі на сьогодні вже готові програмні системи для виявлення і прогозування шахрайських транзакцій, пошуку викидів і відхилень в даних, а також функції статистичного аналізу в цих системах. Розглянуто робочі вікна цих систем, ціни на обслуговування та можливість інтеграції в існуючий бізнес.

Серед розглянутих методів виявлення аномалій в даних, найкращі характерестики мають випадковий ліс та k найближчих сусідів — тому вирішено застосувати ці методи для розв’язання поставленої задачі. Перевагами цих методів є мультикласова класифікація, відносно висока точність та стійкість до викидів в даних.

РОЗДІЛ 2
МАТЕМАТИЧНІ ОСНОВИ ПРОЄКТУ

За кожним алгоритмом машинного навчання стоїть звичайна математика та алгебраїчні обчислення. Навіть при використанні глибокого навчання все зводиться до обчислення матриць та перетворень між ними. В цьому розділі ми розглянемо необхідні математичні поняття для обробки даних та прогнозування аномальних транзакцій.

2.1 Методи препроцесінгу сирих даних

Після того як сирі дані завантажені, обов'язково потрібно перевірити, чи є цільова зміна (якщо це задача класифікації, а не регресії) збалансованою, тобто чи маємо ми однакову кількість як нормальних так і шахрайських транзакцій. Наступним етапом потрібно виконати попередню обробку отриманих даних для начання.

В нашому датасеті дані взято з реального бізнесу, а тому цільова зміна не збалансована. Також з метою збереження конфіденційності даних, оригінальні назви полів, їх функції були зашифровані. Отже, неохідно виконати нормалізацію даних, а також вибрати множину факторів на яких буде проводитися навчання вибраних моделей. Для вирішення даних підзадач вирішено використати такі методи:

1. Нормалізація даних
2. Використання кореляції Пірсона

2.1.1 Нормалізація даних

Для проведення процесу нормалізації найкращим рішенням вважається метод головних компонент. Головна суть і цінність даного методу полягає в зменшені розмірності вхідної вибірки, але при цьому не втрачається основна інформаційна значущість.

Основна ідея методу методу головних компенет заключається в тому, що відбувається заміна k - мірної випадкової величини на m – мірну (m < k), але при цьому втрати інформації дуже низькі. Втрата інформації – це залежність

збереження інформативності функцій від степення, який дозволяє скорочувати та відновлювати дані до початкового стану. Розглянемо один з методів, який базується на m лінійних функціях, що дозволяє такі перетворення. Отже, найкращим підходом, який відомий сьогодні, є метод вибору m головних компонент.

Розглянемо поняття «головна компонента». Допустимо, що на площині задано k-ознак $X_1, X_2 \dots X_k$. Перша головна компонента – це Y_1, що дозволяє зберегти відстань між точками ознак на площині:

$$Y_1 = a_{11}X_1 + \cdots + a_{kk}X_k$$

де $a_{11} \dots a_{k1}$ – коефіцієнти, які обрані так, щоб дисперсія була максимальна:

$$D(Y_1) = \lambda 1$$

Це дає змогу бути впевненим в тому, що об'єкти будуть відрізнятися найбільш сильно за першою компонентою. Щодо другої, то вона також є лінійною функцією вхідних даних:

$$Y_2 = a_{12}X_1 + \cdots + a_{k2}X_k$$

де $a_{11} \dots a_{k2}$ – коефіцієнти, які обираються таким способом, щоб сформовані компоненти Y_1 та Y_2 не корелювали між собою, а її дисперсія $D(Y_2) = \lambda 2$ – має набувати максимального значення серед лінійних комбінацій, які не мають кореляції з Y_1. Отже, друга компонента має відображати найновішу інформацію, яка ніяким чином не пересікається з інформацією першої компоненти.

За тим же алгоритмом знаходять решту компонент, що можна представити наступною формулою [14, 15]:

$$Y_j = \sum_{i=1}^{k} a_{ij}X_i \, , j = \cdots k$$

На рис. 2.1 зображено алгоритм роботи головних компонет:

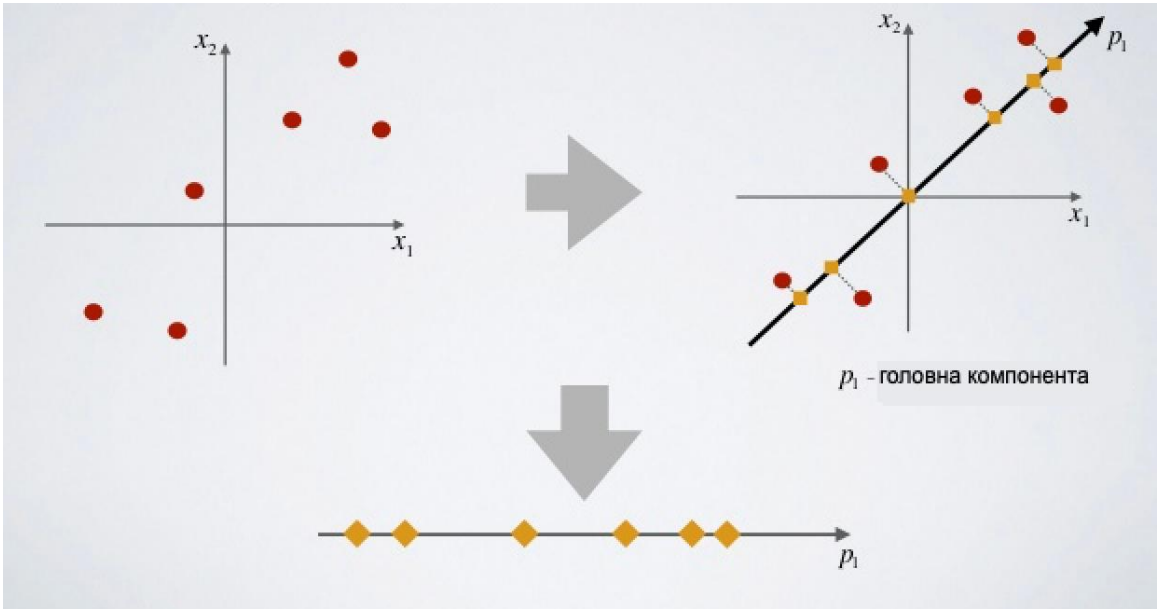


Рис. 2.1. Алгоритм роботи головних компонет [16]

2.1.2 Кореляція Пірсона

В даній задачі використаємо кореляційну матрицю Пірсона, коефіцієнти якої дають змогу показати існуючі лінійні зв’язки між двома факторами. Простими словами, за допомогою матриці Пірсона, ми знатимемо точно дві важливі характеристики між двома факторами:

- Чи є зв’язок між двома факторами.
- Який це зв’язок (примий, зворотній).

Приклад застосування кореляції Пірсона приведений на рисунку 2.2:

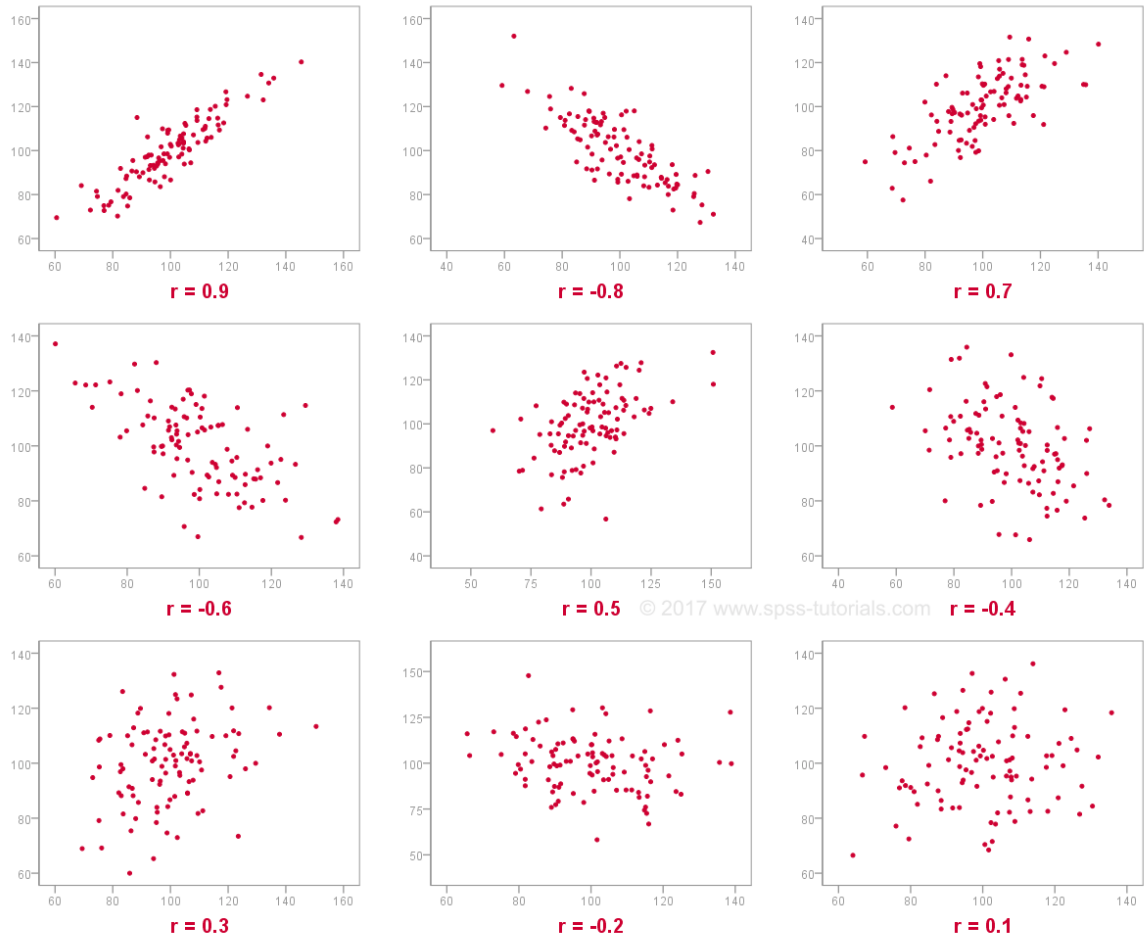


Рис.2.2. Кореляція Пірсона

Припустимо є дві вибірки $x^m = (x_1, \dots x_m)$ та $y^m = (y_1, \dots y_m)$, коефіцієнти кореляції Пірсона можна розрахувати за такою формулою:

$$r_{xy} = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{x})^2 \sum_{i=1}^{m}(y_i - \bar{y})^2}} = \frac{cov(x,y)}{\sqrt{s_x^2 s_y^2}},$$

де:

$\bar{x}, \bar{y}$ – вибіркове середнє x^m та y^m,

s_x^2, s_y^2 – вибіркові дисперсії.

$$r_{xy} \in [-1; 1]$$

Коефіцієнт кореляції Пірсона вказує також на щільність досліджуваного зв'язку:

$|r_{xy}| = 1 \rightarrow x, y$ лінійно залежні,

$|r_{xy}| = 0 \rightarrow x, y$ лінійно незалежні [15].

2.2 K–найближчих сусідів (KNN)

Алгоритм k – найближчих сусідів використовують для задачі класифікації об’єктів. Припустимо, що в нас є m дослідів і кожен з них має своє окреме значення. Всі ці значення відносяться до конкретного одного класу. Задача полягає у визначені одного з цих класів для нерозміченного об‘єкту. Для виконання задчі спочатку зобразимо всі дані на площині (Рис. 2.3).

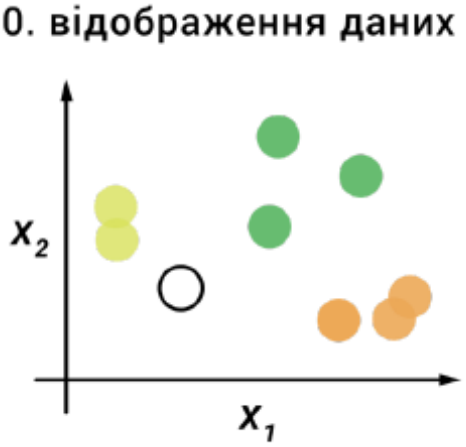


Рис. 2.3. Зображення даних вибірки

Спочатку необхідно задати кількість найближчих сусідів, які будуть використані в розранках для кожного об’єкту – число k. Обов’язкою умовою є виконання правила k > 1.

Наступний етап – це знаходження k записів, які розташовані на мінімальній відстані до вектора ознак іншого сусіда. Для функції розрахунку мають виконуватися наступні вимоги:

- $d(x, y) \geq 0, d(x, y) = 0$ тоді і тільки тоді, коли $x = y$,
- $d(x, y) = d(y, x)$,
- $d(x, z) \leq d(x, y) =$

  $d(y, z)$ за умови, що x, y, z лежать на одній прямій.

x, y, z – вектори ознак об’єктів, які порівнюються. Після впорядкування значень потрібно обчисли Евклідову відстань за формулою:

$$D_E = \sqrt{\sum_{i}^{N} (x_i - y_i)^2},$$

де N – кількість атрибутів у вибірці.

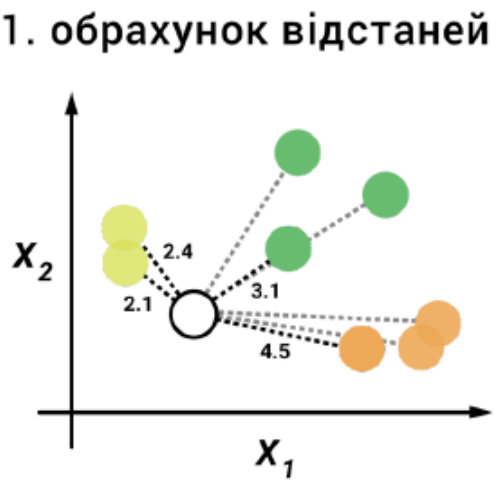


Рис. 2.4. Обрахунок відстаней

Після очислень відстаней, проводимо пошук заданої кількості сусідів (Рис. 2.5).

Рис. 2.5. Пошук сусідів

Наступний етап – проведення голосування і вибір за мажоритарним правом (Рис. 2.6).

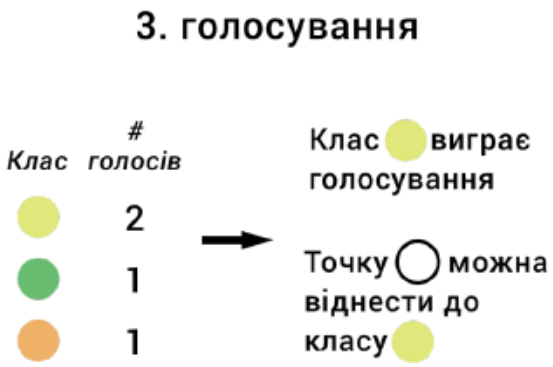


Рис. 2.6. Голосування

Голосування і вибір майбутнього класу виконують за формулою:

$$votes(class) = \sum_{i=1}^{n} \frac{1}{d^2(X,Y)}$$

де $d^2(X,Y)$ – квадрат відстані між відомим об'єктом Y_i і новим X, n – кількість наперед відомих об'єктів класу для якого розраховуються голоси, class – назва класу [17].

2.3 Випадковий ліс (Random Forest)

Ще одним способом пошуку викидів у даних є алгоритм випалкового лісу. Але перед розглядом даного алгоритму слід ввести таке поняття як «бегінг».

Припустимо, у нас є певний метод машинного навчання $\mu(X)$. Необхідно розробити на основі метода $\mu(X)$ новий метод $\tilde{\mu}(X)$, який генеруватиме випадкову підмножину $\overline{X}$ з використанням процедури бутстрапу і подаватиме значення і параметри цієї підмножини на вхід метода
μ: $\tilde{\mu}(X) = \mu(X)$.

Бутстрапінг — це процес вибору випадковим чином l об'єктів з вибірки з їх наступним повернененням назад, тому буде ситуація, за якої певні об'єкти можуть бути вибрані декілька разів, а інші об'єкти можуть бути взагалі не задіяні в процесі навчання. Отже, після розміщення декількох підвибірок одного і того ж об'єкта в бутстрапінговій множині з співставленням ваги про цей об'єкт, ми отримаємо, що даданок при цьому об'єкті з'явиться декілька

разів, а тому і штраф помилки для нього більший. Основна суть методу бегінгу полягає в тому, що нам потрібно навчити не один алгоритм, а певну кількість – $b_n(x)$.Дані алгоритми будуються за допомогою раніше визначеного методу $\tilde{\mu}$. Вкінці отримуємо вихідну композицію, яка розрахована на основі середніх значень з базових алгоритмів:

$$a_N(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(x) = \frac{1}{N}\sum_{n=1}^{N} \tilde{\mu}(X)(x)$$

В основі алгоритму випадкового лісу лежить бегінг для дерев рішень. Важливо відмітити, що в алгоритмі відбувається випадковий вибір за об'єктами та ознаками, а тому між базовими моделями дерев рішень взаємозв'язок зменшується. Спершу формується бутстрапівська вибірка і модель навчається на цій вибірці, а потім при навчанні знову відбувається випадковий вибір підмножини ознак.

В даному алгоритмі ознака для розподілу визначається з рандомної підмножини m. Прийнято для вирішення задачі класифікації обирати $m = \sqrt{|d|}$, для регресійних — $m = \left[\frac{d}{3}\right]$, де d – кількість ознак в датасеті.

Для задач класифікації необхідно будувати дерево рішення до моменту, коли на листі буде не більше одного об'єкта. Це є передумовою більш точного визначення класу, але не необхідною умовою.

Для начання кожного дерева використовується нова підмножинна об'єктів, тому для конкретного дерева зразки даних, які не були вибрані для начання слугують для обчислення out-of-bag помилки:

$$OOB = \sum_{i=1}^{l} L(y_i \frac{1}{\sum_{n=1}^{N}|x_i \notin X_N|}\sum_{n=1}^{N}|x_i \notin X_N|\, b_n(x_i))$$

Де L(y,z) – функція втрат. При достатньо великій кількості N розрахована оцінка наближається до leave-one-out-оцінки, але є набагато простіша в обчислені [18].

2.4 Методи оцінки точності

Для представлення результатів застосування моделей машинного начання в бізнесі зазвичай використовують робочу характеристику кривої. Але дана метрика дуже чутлива до незбалансованих вибірок, а тому не зовсім релевентна в оцінці точності прогнозу.Для задач з незбалансованими даними рекомендовано використовувати криву Precision-Recall (PR). Вона дає набагато глибше розуміння того як прогнозуються нові класи [19].

Отже, як ми визначили – точність є не ідеальною характеристикою моделі при незбалансованих даних і цілком покладатись на неї не варто. Тому, варто ввести інші метрики оцінки, які б дали змогу оцінити якість моделі:

1. Матриця помилок (confusion matrix) – таблиця, перетин рядків і стовпців якої показує кількість правильно і неправильно визначених об’єктів.
2. Точність – відсоток правильно визначених об’єктів серед усіх об’єктів, позначених як позитивні. Точність є важливою метрикою, оскільки нам потрібно розуміти, чи класифікатор не дає багато хибних позитвних результатів.
3. Повнота – співвідношення кількості правдиво визначених об’єктів та сумарної кількості позитивних об’єктів в тестовому датасеті. Дана метрика вказує наскільки «чутливим» є класифікатор. Також вона дає зрозуміти наскільки повно ми вгадуємо об’єкти певного класу. Висока точність підтверджує, що ми покрили більшість об’єктів з цього класу.
4. F1-оцінка: усереднений показник між точністю та повнотою.

Припустимо є певний датасет, в якому знаходяться дані тільки двох класів (бінарна класифікація) та навчена модель для прогнозування класу на нерозмічених даних. Після визначення прогнозу для даних можна обчислити матрицю помилок. Для бінарної класифікації вона матиме вигляд як на рис. 2.7.

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Рис. 2.7. Матриця помилок

На Рисунку 2.7, $\hat{y}$ – значення класів, які були спрогнозовані алгоритмом, а y – значення, взяті з тестового датасету, тобто наперед відомі правидьні класи. Звідси стає зрозуміло, що існує два типи помилок при розв'язані задач класифікації:

1. Хибно негативні (False Negative, FN)
2. Хибно позитивні (False Positive, FP)

Для визначення частки правильних відповідей (як позитивних так і негативних) використовуються метрику accuracy, яка розраховується за формулою:

$$accuracy = \frac{1}{l}\sum_{i=1}^{l}[a(x_i) = y_i]$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Дана метрика є найпростішою при оцінці натренованої моделі, а також легкою для сприйняття [20].

Для розранку точності натренованої моделі застосуємо наступну формулу:

$$precision = \frac{TP}{TP + FP}$$

Метрика точності дозволяє нам зрозуміти, який відсоток об'єктів, що були класифіковані як позитивні, є справді позитивними.

Для обчислення метрики повноти застосуємо наступну формулу:

$$recall = \frac{TP}{TP + FN}$$

Метрика повноти дозволяє нам оцінити класифікатор в контексті того, наскільки багато обє'ктів певного класу було правильно спрогнозовано.

Варто відмітити, що за умови $a(x) = [p(x) > t]$, і при збільшені певного порогу t, точність моделі буде вищою, а от повнота зменшуватиметься.

Також хорошою характеристикою точності і повноти є те, що вони не корелюють з розмірами класів у датасеті, тобто дані метрики чудово працють при незбалансованих вибірках і коректно дають оцінку алгоритму.

Для полегшення інтепретування результатів введено декілька понять, які об'єднують метрики точності і повноти. Найпопулярнішою і зрозумілою є метод визначення F-міри – це усереднене значення між повнотою і точністю. Обчислити F-міру можна за формулою:

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Введення і застосування поняття F-міри є набагато інформативнішим, ніж наприклад простого обчислення середнього арифметичного між цими двома величинами: середнє гармонічне є чутливим при наближені однієї з метрик до нуля, а тому і все значення буде прямувати до нуля.

Ще однією поширеною метрикою обчислення якості роботи моделі є застосування площі, яка знаходиться під ROC-кривою. Метрика відома як Area Under ROC Curve, або AUC-ROC. Для обчислення даної метрики неохідно ввести двовимірну площину, в якій коорддинатними осями будуть частки правильно (True Positive Rate, TPR) і неправильно (False Positive Rate, FPR) спрогнозованих об'єктів. Частки можна визначити за формулами:

$$FPR = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{TP + FN}$$

Кожен поріг величини t відповідає конкретній точці на цій двовимірній площині. Сумарно існує l + 1 таких порогів. Максимальне значення $t_{max} = \max_i b(x_i)$ може бути досягнути лише за умови TPR = 0, FPR = 0. Мінімальне $t_{min} = \min_i b(x_i) - \varepsilon$ може бути досягнуто відповідно при TPR = 1, FPR = 1.

ROC-крива – це крива, що побудована в двовимірному просторі, при послідовному з'єднані точок з порогом $b(x_1) - \varepsilon, b(x_1), b(x_2), \ldots, b(x_l)$ від початкової точки (0, 0) до кінцевої (1, 1).

Площа, що знаходиться нижче утвореної лінії, називається AUC-ROC. Вона приймає значення від нуля до одиниці. Якщо ми застосуємо в якості моделі генератор випадкових значень, то AUC-ROC при великій кількості об'єктів буди наближатися до значення 0.5. [21]

Приклад ROC-кривої наведено на рис. 2.8.

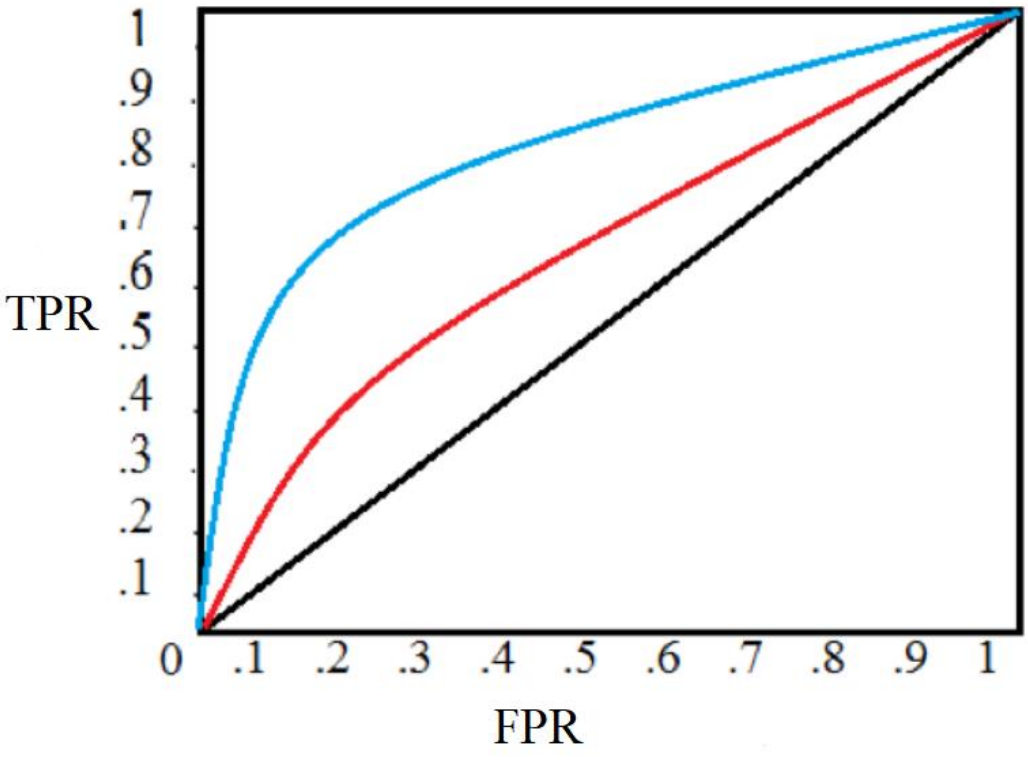


Рис. 2.8. Приклад ROC-кривої.

ВИСНОВКИ ДО РОЗДІЛУ 2

Оскільки тренувальна вибірка для заданої задачі є незбалансованою, то виникає проблема роботи з незбалансованими даними для тренування моделі. Було розглянуто методи роботи з такими даними, а також їх перетворення до нормальної форми. Конкрето було розглянуто метод головних компонент – як один з кращих методів нормалізації даних перед запуском навчання а також досліджено метод знаходження взаємозв'зків у даних – матриця Пірсона. Знаходження кореляції між факторами є важливим кроком у розробці моделі, оскільки це дозволяє видалити зайві ознаки з даних, які мало впливають на цільову зміну, і відповідно оптимізувати процес начання. Особливо важливо для алгоритму KNN.

Також було розглянуто основні метрики за якими відбувається оцінка та інтерпретація результатів і точності моделі. В цій роботі для оцінки роботи моделі будуть використані такі з них: confusion matrix, recall, precision, ROC-AUC.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ТА РОЗРОБКА МОДЕЛІ

В даному розділі ми спректуємо та розробимо модель, спираючись на один із вибраних алгоритмів. Виконаємо покроково всі етапи роботи з данимим від завантаження сирих файлів до готового файлу із спрогнозованим класом.

3.1. Структура системи виявлення відхилень в транзакціях

Отже, спершу розглянемо нашу систему обробки даних і начання моделі в цілому (рис.3.1), а потім перейдемо до кожного модуля окремо. До системи обробки і начання даних входять такі компоненти:

1. Модуль передобробки сирих даних
2. Модуль розділення історичних даних на тренувальний і тестувальни датасет
3. Модуль навчання і вибору кращого алгоритму для кінцевої моделі пошуку викидів у даних

На етапі процесу передобробки даних (ETL – Extract, Transform, Load) обов’язково виконують такі перевірки:

1. Перевірка на збалансованість цільової змінної
2. Перевірка наявності дуплікатів в даних
3. Перевірка на пропуски в даних

При виявлені незбалансованості в датасеті –необхідно виконати процес нормалізації даних. І тільки після цього виконуємо розділення на тренувальну та тестову підвибірки. Якщо нормалізуються тренувальні дані, то обов’язково необхідно нормалізувати і тестові, а потім і на проекті дані.

Наступним етапом є застосування вибраних алгоритмів до оброблених даних. У вирішенні задачі вибрано наступні алгоритми:

1. Випадковий ліс
2. K найближчих сусідів
3. Метод голосування (Випадковий ліс+K найближчих сусідів)

Точність алгоритмів порівнюємо з використанням раніше визначених метрик.

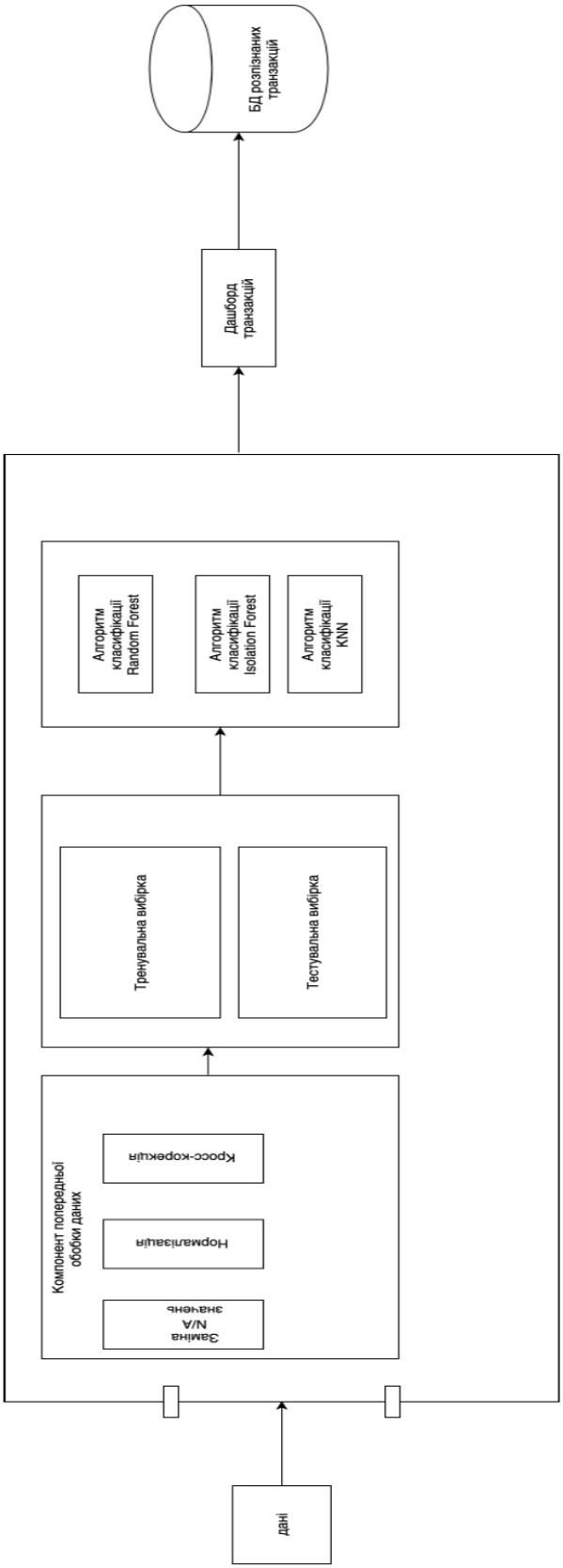


Рис. 3.1. Структура системи

3.2 Розробка системи

Для розробки системи визначення шахрайських транзакцій було обрано скриптову мову програмування – Python. Сьогодні це найпопулярніша мова для написання скриптів для збору, передобробки, аналізу даних, а також для написання програмних модулів для навчання прогнозних моделей.

До явних переваг застосування Python можна віднести наступні:

1. Проста у вивченні та використані. Багато функцій вже написано в готових модулях, а назви дозволяють швидко розібратись у функціоналі.
2. При використанні бібліотек для візуалізації (matplotlib, seaborn) Python можна використовувати як хороший і швидкий спосіб візуалізації даних і залежностей в них.
3. Зручна робота з структурованими та неструктурованими даними. Також цілі набори інструментів для роботи з математичними даними –як от бібліотеки Pandas, SkiPy, IPython, NumPy.

Для попередньої оцінки для розробки демо продукту виявлення аномальностей в датасеті, було використано Jupyter Notebook– веб-інтерфейс для оболонки IPython –найбільш відомий і зручний для створення інтерактивних звітів.

Серед значимих переваг данного середовища хотів би виділити наступні:

1. Можливість створювати візуалізації даних декількома строчками коду.
2. Легка взаємодія з сюжетами
3. Додавання до коду коментарів та тексту в форматі Markdown/
4. Легка і зрозуміла взаємодія з різними форматами файлів та базами даних. [22]

3.3 Застосування вибраних алгоритмів

3.3.1 Передобробка даних та загальний огляд

Для навчання моделі виявлення аномальних транзакцій було взято датасет, сформований на історичних даних з 284 807 зразків даних. Розподіл класів несбалансований: в даних присутні лише 492 шахрайські транзакції [23] (Рис. 3.2)

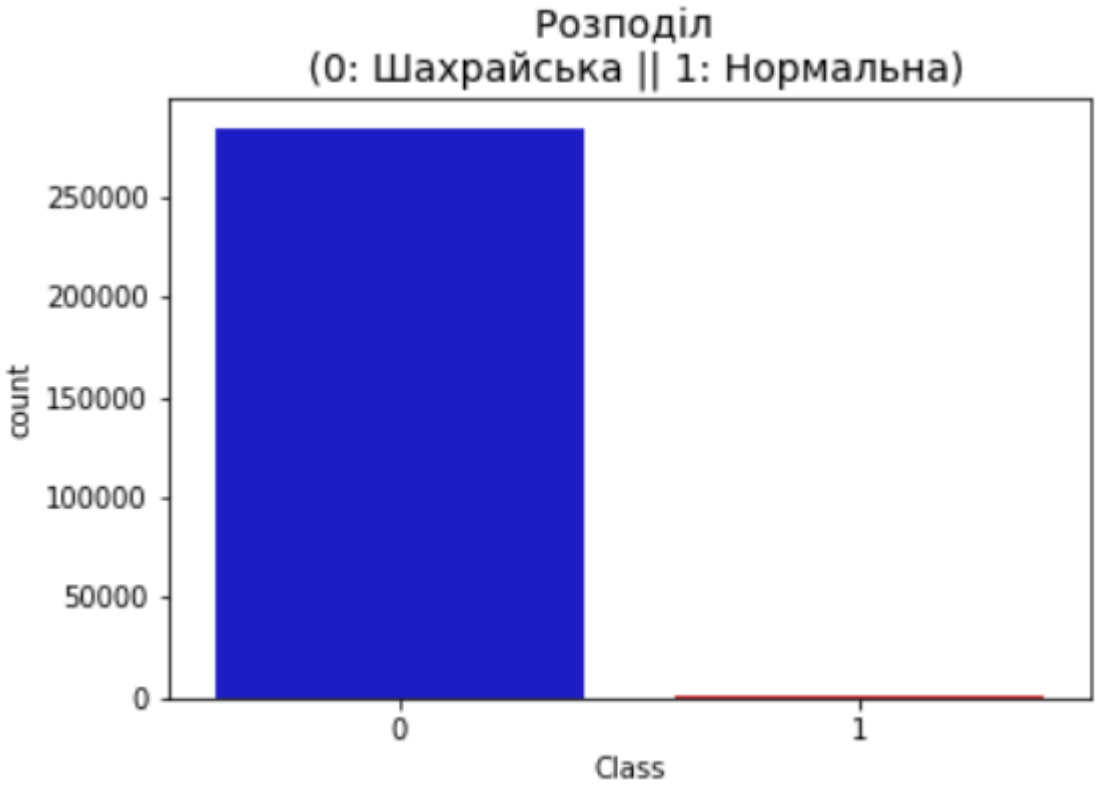


Рис. 3.2. Класовий розподіл у вибраному датасеті

З метою збереження конфіденційності даних та анонімності клієнтів, ознаки були перетворені та зашифровані методом головних компонент. Фактично ми маємо анонімний датасет із ознаками V1, V2, ... V28, де V1, V2, ... V28 –результат роботи PCA. Окрім зашифрованих даних, у наборі є два поля, до яких не було застосовано метод головних компонент: «час», «сума». Ознака часу показує, скільки минуло часу між досліджуваною і першою транзакцією в датасеті. Ознаці «сума» відповідає кількість грошей, на яку проводилася транзакція. Цільова ознака «Class» має бінарний розподіл, 0 або 1, та відповідає типу транзакції:

- 1 — шахрайська
- 0 — нормальна

Оскільки в нас один датасет, то виконаємо його розділення на тренувальну частину та валідаційну –для оцінки роботи моделі. На тренування моделі виділимо 67% від вибірки, на тест –решту.

Зважаючи на те, що в нас в даних є 2 незашифровані поля, ми можемо дослідити чи є між ними якась залежність (рис. 3.3 та рис. 3.4).

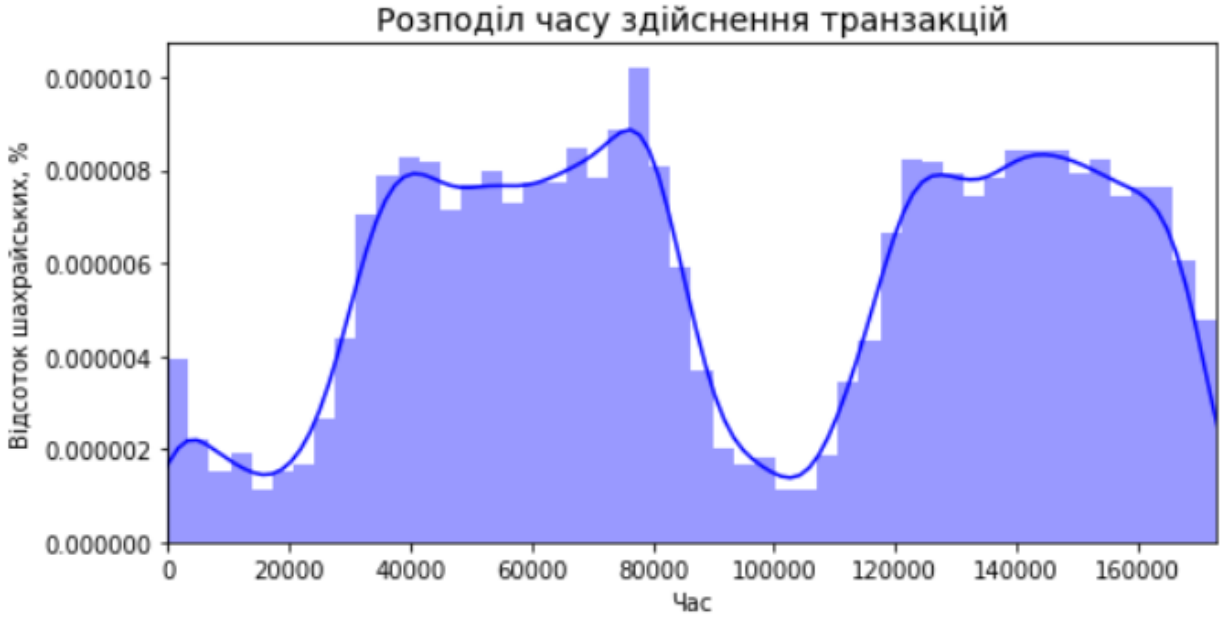


Рис. 3.3. Розподіл кількості шахрайських транзакцій від часу

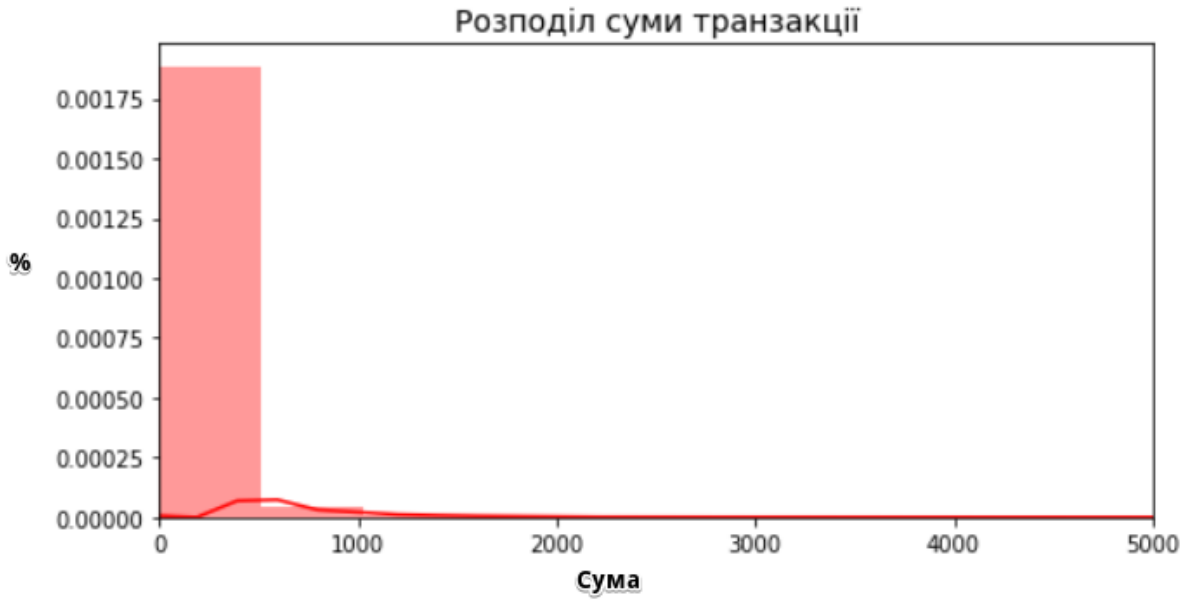


Рис. 3.4. Розподіл суми транзакції

Як видно з рис. 3.3, в залежності від часу, що минув від першої транзакції є два пікові моменти, коли різко виростає відсоток здійснення шахрайських транзакцій. Сума транзакції має нормальний розподіл, суттєвої різниці в сумі між шахрайскою і звичайною немає.

На етапі передобробки даних обов'язково потрібно:

1. Зробити попередній аналіз даних
2. Перевірити наявність дублікатів у вибірці
3. Виконати нормалізацію всіх ознак
4. Знайти матрицю кореляції
5. Оптимізувати кількість кількість даних шляхом видалення слабокорелючих з цільовою зміною
6. Використати алгоритм для збалансування

Отже, виконаємо попередній огляд розподілу класів в просторі (рис. 3.5)

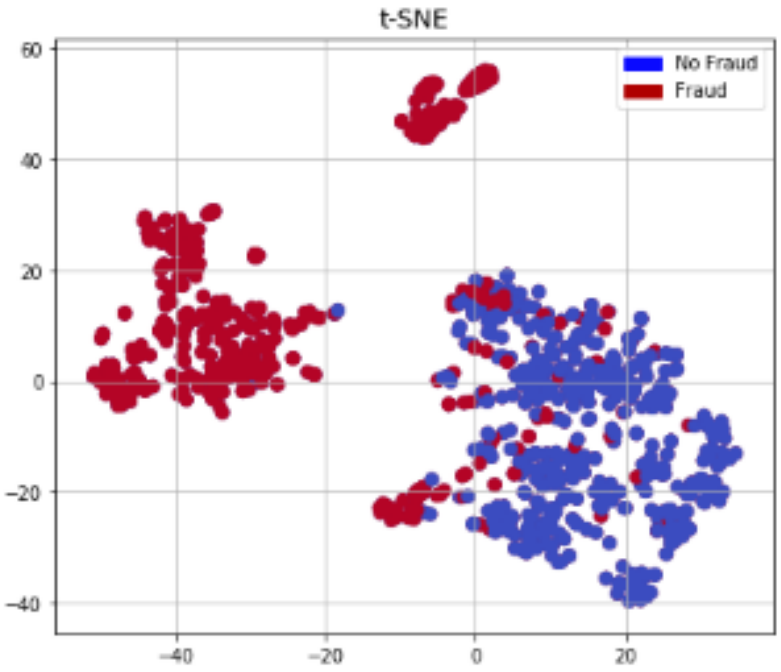


Рис. 3.5. Розподіл двох класів у даних

З рис. 3.5 видно, що шахрайські транзакції мають дві чітко виражені групи та невелику кількість, які прилягають до класу нормальних.

Наступний етпом обробки даних для алгоритму є видалення дублікатів в даних. Виконаємо перевірку на наявеість дублікатів в кожному класі (рис. 3.6). Як видно з перевірки, дублікати зустрічаються як серед нормальних транзакцій, так і серед шахрайських. Оскільки дані не збалансовані, то видалимо задубльовані строки лише в класі нормальних транзакцій.

```
In [13]: print('Class type : rows duplicated \n',
               data.loc[data.duplicated(), 'Class'].value_counts())
```

```
Class type : rows duplicated
 0    1062
1      19
Name: Class, dtype: int64
```

Рис. 3.6 – Визначення кількості дублікатів в даних

На третьому кроці виконуємо нормалізацію даних. Особливо важливо виконувати даний крок, якщо планується застосовувати алгоритми чутливі до викидів у даних. Оскільки, для збереження анонімності в даних, датасет містить вже сформовані дані одного виду після застосування методу головних компонент, то необхідно також виконати нормалізацію ознаки «Amount», застосувавши алгоритм StandartScaler (рис. 3.7).

```
In [21]: std_scal = StandardScaler()

normed_amount = std_scal.fit_transform(
    data['Amount'].values.reshape(-1, 1))
data['Amount'] = normed_amount
print(
'Normalized "Amount" feature stats:\n',
data['Amount'].describe()
)
```

```
Normalized "Amount" feature stats:
 count    2.848070e+05
mean     3.202236e-16
std      1.000002e+00
min     -3.532294e-01
25%     -3.308401e-01
50%     -2.652715e-01
75%     -4.471707e-02
max      1.023622e+02
Name: Amount, dtype: float64
```

Рис. 3.7. Поле “Amount” після роботи алгоритму StandartScaler

Ще одним важливим етапом розробки моделі є перевірка даних на наявність пропусків і, за наявності, видалення таких зразків з набору, або краще заміна іншими значеннями. Наприклад середнім по колонці, медіаною, а центром класу. Перевірка нашого датасету показала, що в нас дані повністю заповнені (рис 3.8).

```
In [3]: transactions_df = pd.read_csv('creditcard.csv', header=0)
display(
'Found {0} null values in transactions dataset'.format(
transactions_df.isnull().sum().sum()
)
)
```

```
'Found 0 null values in transactions dataset'
```

Рис. 3.8. Наявність дублікатів в даних

Після нормалізації потрібних ознак для начання переходимо до оптимізації і скорочення часу начання. Найперше знаходимо кореляції між ознаками за допомогою матриці Пірсона (рис. 3.9). Після визначення взаємозв’язків (прямих чи обернених) в даних, можна буде видалити

корелюючі між собою ознаки. З матриці видно, що в даних є ознаки, які корелюють між собою та декілька ознак, що помітно корелюють з сумою та часом транзакції.

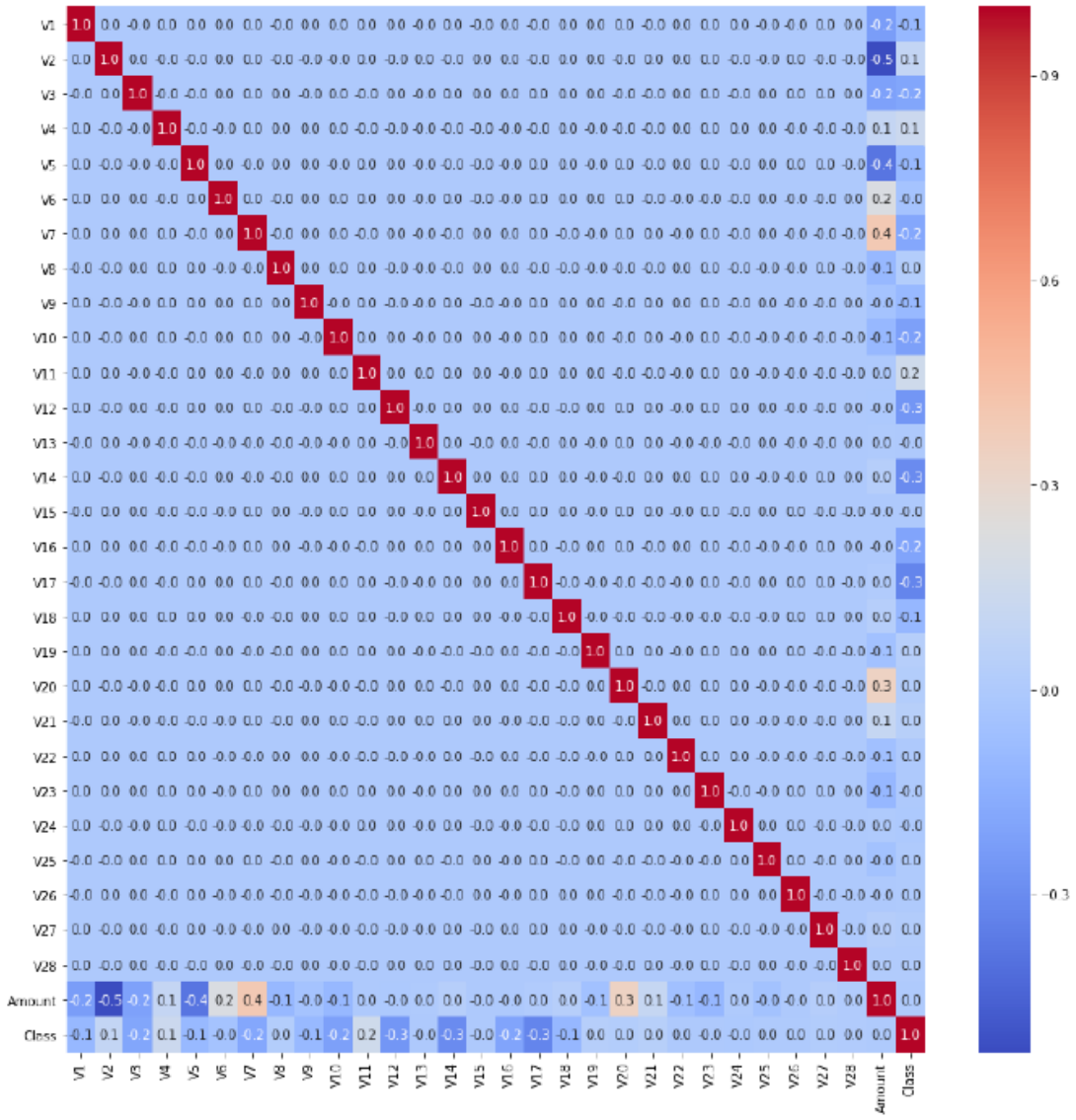


Рис. 3.9. Кореляційна матриця Пірсона

Після дослідження взаємної кореляції між ознаками, варто окремо дослідити зв’язок з цільовою змінною (рис. 3.10).

```
V22       0.000805
V23       0.002685
V25       0.003308
V15       0.004223
V26       0.004455
V13       0.004570
Amount    0.005632
V24       0.007221
V28       0.009536
V27       0.017580
V8        0.019875
V20       0.020090
V19       0.034783
V21       0.040413
V6        0.043643
V2        0.091289
V5        0.094974
V9        0.097733
V1        0.101347
V18       0.111485
V4        0.133447
V11       0.154876
V7        0.187257
V3        0.192961
V16       0.196539
V10       0.216883
V12       0.260593
V14       0.302544
V17       0.326481
Class     1.000000
Name: Class, dtype: float64
```

Рис. 3.10. Кореляція з цільовою змінною

Після дослідження кореляції між ознаками та цільовою змінною видно, що найсильніше на віднесення транзакції до того чи іншого класу впливають ознаки: «V17», «V14», «V12», «V10», «V16», «V3», «V7», «V4».

Інші ознаки, які присутні в наборі даних, набагато слабше впливають на кінцеве рішення про віднесення транзакції до шахрайської. Отже, можна впевнено виключати ці параметри з даних при навчанні моделі, оскільки це дозволить звільнити додаткові ресурси, або ж зменшити існуючі втрати – особливо корисно, коли модель в режимі реального часу перенавчається на історичних даних перед здісненням прогнозу. Видалення зайвих ознак сильно

пришвидшує роботу алгоритмів, які використовують всі дані для начання, наприклад вибраний алгоритм k– найближчих сусідів.

Як вже було визначено раніше, дані для начання моделі незбалансовані, а тому рекомендовано використати якийсь алгоритм для збалансування класів. Наприклад, до незбалансованих даних можна застосувати один з цих алгоритмів:

- SMOTE
- Undersampling

Але оскільки в нашому наборі даних дуже мало записів про шахрайські транзакції, то варто застосувати метод перетворення SMOTE (метод передискретизації синтезованих меншин), тобто даний алгоритм створює нові, неіснуючі раніше зразки записів, які максимально схожі на елементи класу, якій перебуває в меншості, порівняно з іншим.

Метод SMOTE працює за наступним алгоритмом: для синтезу нового елементу в даних необхідно знайти різницю:

$$d = X_b - X_a,$$

де X_a, X_b — вектори ознак елементів, що знаходяться найблежче до досліджуванного.

Індекси a і b визначаються застосуванням алгоритму k – найближчих сусідів до меншого класу. Необхідно і достатньо для певного елемента a знайти підмножину сусідів k, з якої залишиться потім тільки елемент b.

Наступним кроком є знаходження $\hat{d}$, яке отримують домножуванням d на випадково згенероване число на проміжку від нуля до одиниці. Тоді вектор ознак нового синтезованого елементу можна обчислити, додавши $\hat{d}$ до X_a [24].

Приклад роботи методу наведено на рис. 3.11.

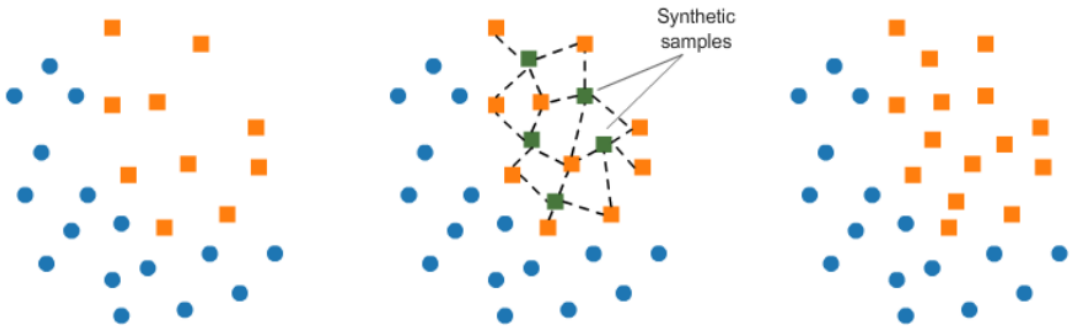


Рис. 3.11. Приклад роботи методу SMOTE [24]

Суть методу undersampling-у полягає у видаленні певних зразків даних з мажоритарного класу. Завдяки цьому можна залишити, випадковим чином вибраних, таку кількість зразків як і в міноритарному класі. Але із застосуванням цього методу варто бути особливо обережним — є ризик видалення і відповідно втрати цінної інформації для навчання. Суть методу наведено на Рис. 3.12 [24].

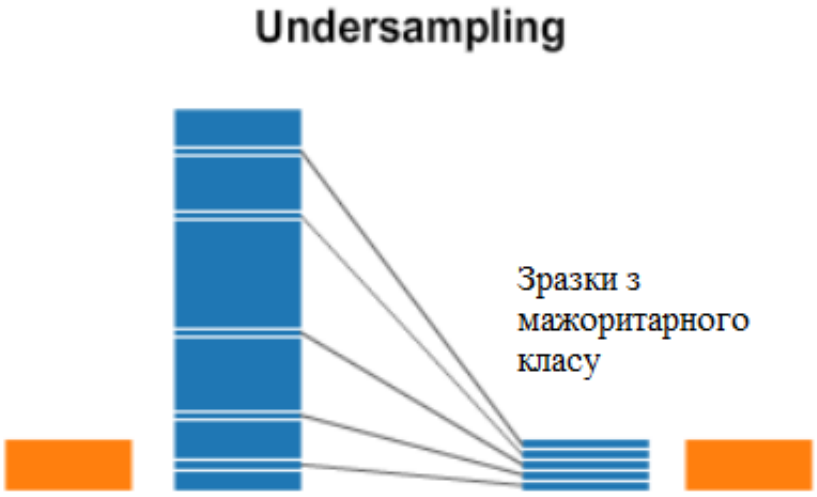


Рис. 3.12. Результат роботи методу undersampling [24]

3.3.2 Вага ознак та інформаційна цінність

Для розв’язання завдання вибору важливих ознак було використано параметри WoE (Weight of evidence) та IV (Information Value). Для обрахунку IV в модулі preprocesing.py були описані наступні функції:

- get_range_woe_iv — функція для перетворення неперервних ознак в дискретні діапазони
- recursive_get_range_woe_iv — рекурсивна функція для визначення інформаційної цінності для кожного визначенного діапазону
- show — функція виводу інформації

Результат застосування визначення інформаційної цінності наведено на рис 3.13. Якщо значення коефіцієнту менше ніж одна сота, то прийнято такі ознаки видаляти із датасету. В нас всі ознаки більше цього порогового числа.

	feature_name	maximized_IV
0	V1	1.184134
1	V2	1.777144
2	V3	3.330727
3	V4	4.313076
4	V5	0.709378
5	V6	0.013202
6	V7	1.111598
7	V8	1.046572
8	V9	2.581701
9	V10	4.084238
10	V11	5.423019
11	V12	6.186089
12	V13	0.049738
13	V14	6.919044
14	V15	0.024951
15	V16	4.526210
16	V17	6.001780
17	V18	3.398168
18	V19	0.968142
19	V20	0.073019
20	V21	0.230901
21	V22	0.096025
22	V23	0.030907
23	V24	0.121093
24	V25	0.126251
25	V26	0.021853
26	V27	0.224254
27	V28	0.179646
28	Amount	0.000002
29	Class	0.000000

Рис 3.13. Коефіцієнти інформаційної цінності

3.3.3 Застосування підходу API

Оскільки передбачається можливість використання системи в режимі реального часу, то найзручнішим підходом до розповсюдження рішення є використання функціоналу API. Загальна схема роботи зображена на рис. 3.14. Для розробки функціоналу використаємо веб-фрейморк Flask. Даний фреймворк дозволяє легко розгортати невеликі додатки із мінімальними ресурсами.

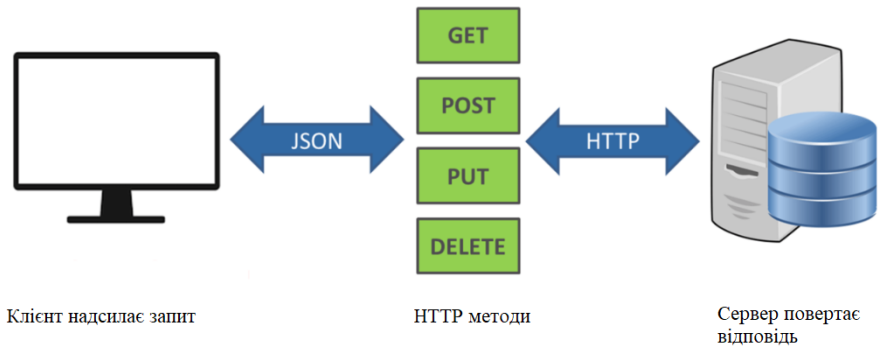


Рис. 3.14. Схема роботи через API

Для розгортання системи на сервері, створено модуль app.py. Для коректної роботи в модулі написано функції для завантаження натренованої моделі, читання отриманого json файлу із даними, які передаються під час здійснення транзакції, переформатування даних та повернення результатів до клієнта.

Після того, як транзакція сформована, в систему надходить 29 параметрів. Спочатку ці параметри формуємо в словник, далі на основі словника створюємо JSON файл, який буде переданий як аргумент методу POST. При виклику методу POST першим аргументом передаємо адресу розгонутого додатку, другим аргументом – файл із даними.

На стороні сервера визначаємо функцію для періодичного оновлення і отримання JSON файлу, сформованого клієнтом. Як тільки сервер отримує файл, відбувається його розпаковка у словник, оскільки клієнт може надіслати відразу декілька транзакцій в одному файлі. Після парсингу файлу із словника створюється датафрейм –двовимірний масив даних. До отиманого масиву застосовуємо імпортовану перед цим модель та отримуємо прогноз.

Після отримання рішення про клас транзакції, упаковуємо відповідь у такий же JSON файл та передаємо назад клієнту. Клієнт парсить файл та приймає рішення чи віддавити транзакцію на процесинг банку.

На стороні сервера видно зчитування, переформатування та створення двовимірного масиву для прогнозу. Приклад роботи з даними на сервері

3.3.4 Розробка класу для вибору моделі

Для автоматизації тестування і вибору моделі із найкращими результуючими метриками розроблено модуль test_class.py. В модулі визначено функції чотири функції:

- __init__ – функція конструктор, відповідає за ініціалізацію моделі, отриманння даних із тренувальної та тестувальної вибірки, а також прапора, що визначає чи передано розширені параметри для навчання.
- fit_model — функція для навчання конкретної моделі
- show_metrics — функція для розрахунку всіх вибраних метрик оцінки точності прогнозування
- plot_confusion_matrix — функція, що приймає на вхід реальні правдиві значення класу та тільки що спрогнозовані і будує матрицю.

Перед викликом методу ініціалізації класу, необхідно за потреби створити список параметрів, які повинні бути відтестовані на цій моделі. Наприклад для алгоритму Random Forest, це може бути кількість дерев при начані, чи кількість листків.

3.3.5 Використання Random Forest

Після закінчення підготовки даних можна перейти до начання і тестування моделей. Спершу застосуємо алгоритм random forest. Даний алгоритм працює за наступними кроками:

1. Для n = 1… N необхідно визначити підмножину $\widetilde{X_n}$ із застосуванням методу бутстрапінгу.
2. На основі згенерованої вибірки будується дерево рішення $b_n(x)$:
    - Дерево будується до моменту, поки виконується умова — кількість об’єктів $\leq n_{min}$.
    - При будь-якому розбиті обирається m випадково вибраних ознак з p, і оптимальна підмножинна знаходиться лише для цих m ознак.

3. Повертаємо композицію $a_n = \frac{1}{N}\sum_{n=1}^{N} b_n(x)$[25].

Застосувавши алгоритм випадкових лісів до оброблених даних, вдалося розпізнати 97% шахрайських транзакцій, що є дуже хорошим результатом.

Основні метрики оцінки і їх значення алгоритму для random forest:

$$confusion\ matrix = \begin{pmatrix} 136 & 26 \\ 18 & 93806 \end{pmatrix};$$

$$Precision = \frac{TP}{TP + FP} = \frac{136}{136 + 18} = 0{,}883;$$

$$recall = \frac{TP}{TP + FN} = \frac{136}{136 + 28} = 0{,}84;$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot 0{,}88 \cdot 0{,}84}{0{,}88 + 0{,}84} = 0{,}86;$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{136 + 93806}{136 + 93806 + 18 + 26} = 0{,}97;$$

Для кращої інтерпретації результатів зображено ROC-криву (рис. 3.15).

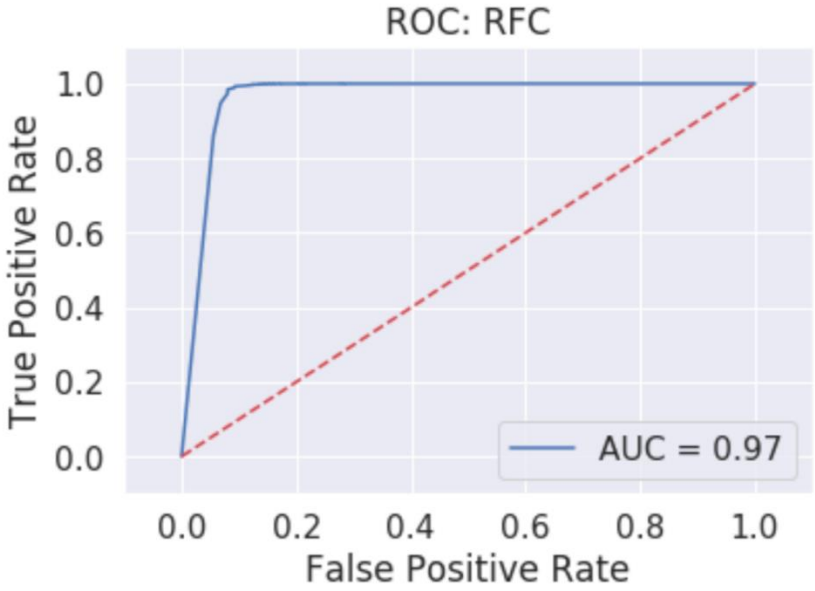


Рис. 3.15. ROC крива для random forest

3.3.3 Використання (KNN)

Наступним тестуємо алгоритм k – найближчих сусідів, який був також обраний як один із способів вирішення задачі в другому розділі.

Етапи роботи алгоритму KNN:

1. Розділення всіх даних трейн і тест вибірки
2. Обчислити відстань до кожного з об’єктів тренувальної вибірки (для обчислення може бути використана не тільки Евклідова відстань, а й, наприклад, Манхетенська, косинусна, критерій кореляції Пірсона)
3. Вибрати K об’єктів начальної вибірки, відстань до яких мінімальна
4. Підраховуємо кількість кожного класу, який з’явився
5. Мажоритарним методом визначаємо клас, до якого варто віднести транзакцію

В результаті застосування алгоритму KNN (K = 5) до оброблених даних вдалось розпізнати правильно 93% усіх транзакцій, проте розпізнавання саме шахрайських транзакцій має низький відсоток. Результати для K-найближчих сусідів:

$$confusion\ matrix = \begin{pmatrix} 136 & 26 \\ 273 & 93551 \end{pmatrix};$$

$$Precision = \frac{TP}{TP + FP} = \frac{136}{136 + 273} = 0{,}33;$$

$$recall = \frac{TP}{TP + FN} = \frac{136}{136 + 26} = 0{,}84;$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot 0{,}33 \cdot 0{,}84}{0{,}33 + 0{,}84} = 0{,}48;$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{136 + 93551}{136 + 93551 + 273 + 26} = 0{,}93;$$

Для кращої інтерпретації результатів зображено ROC-криву (рис. 3.16).

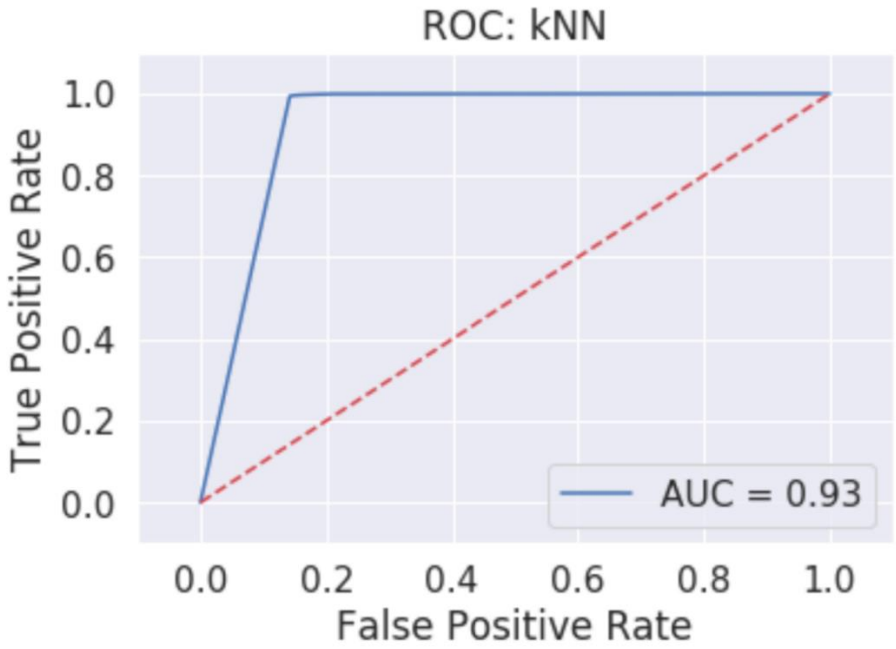


Рис. 3.16. ROC крива для KNN

Варто зауважити, що вибирати алгоритм KNN для систем з дуже великими наборами даних нераціонально, оскільки він має нижчу точність порівняно з іншими алгоритмами, а також низьку швидкість класифікації: якщо при навчанні вибіркаа містила N об’єктів, при прогнозуванні M об’єктів, а простір K-розмірний, то складність обчислень може бути оцінена як O(K*M*N).

3.3.4 Використання Voting model

Часто на практиці доводиться об’єднувати результати двох і більше класифікаторів для підвищення точності. Заради експерименту було об’єднано результати логістичної регресії та випадкових лісів.

В результаті застосування об’єднання алгоритмів до оброблених даних вдалось розпізнати правильно 98% усіх транзакцій. Результати для Voting model:

$$confusion\ matrix = \begin{pmatrix} 97 & 15 \\ 48 & 93471 \end{pmatrix};$$

$$Precision = \frac{TP}{TP + FP} = \frac{97}{97 + 48} = 0{,}67;$$

$$recall = \frac{TP}{TP + FN} = \frac{97}{97 + 15} = 0{,}87;$$

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot 0{,}67 \cdot 0{,}87}{0{,}67 + 0{,}87} = 0{,}76;$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{97 + 93471}{93471 + 97 + 48 + 15} = 0{,}98;$$

Для кращої інтерпретації результатів зображено ROC-криву (рис. 3.17).

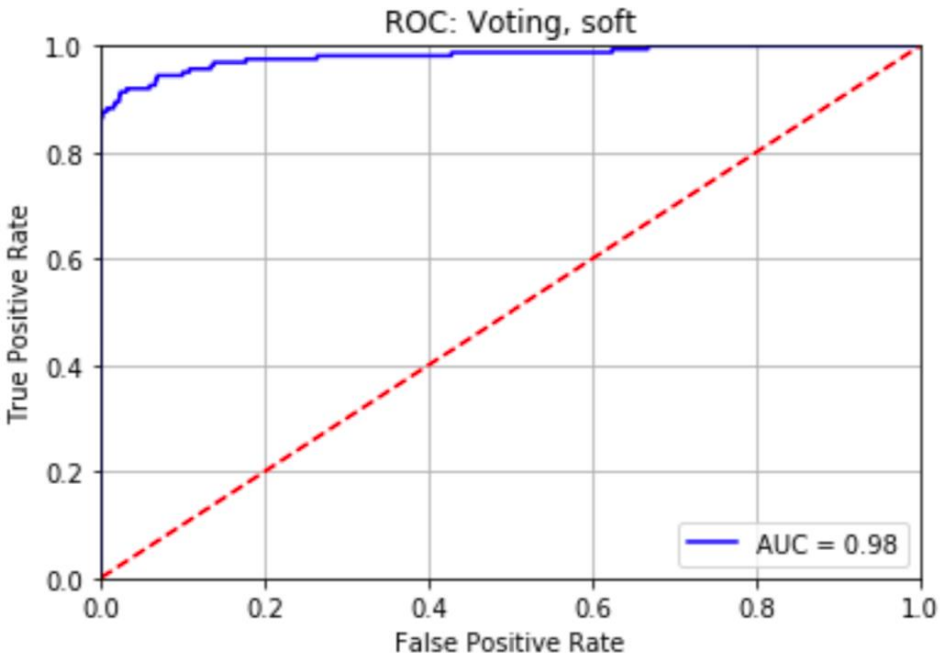


Рис. 3.17. ROC крива для об’єднання алгоритмів

3.4 Способи покращення моделі для застосування в продакшені

3.4.1 Застосування моделі скорингу, а не класифікації

На відміну від алгоритмів класифікації, скорингова модель дозволяє на виході отримати ймовірності того, що певний зразок даних належить до того чи іншого класу. Фактично, ми отримуємо своєрідний «бал», на який можна встановити лінію відсікання і по ній приймати рішення про приналежність до класу шахрайської транзакції. Застосування данного способу є ефективною, коли є чітке розуміння про те, що означають всі ознаки в даних.

3.4.2 Застосування AWS для масштабування

Для мастшабування та підвищення швидкості і точності прогнозу в режимі реального часу рекомендовано застосувати технологію AWS (рис. 3.18).

Алгоритм обробки даних із застосуванням AWS:

1. База даних DynamoDB періодично фіксує і записує зміни.

2. За допомогою AWS Glue відбувається регулярне витягування даних з бази та виконання перенавчання моделі на нових даних з подальшим збереженням моделі до Amazon S3.
3. Також AWS Glue, використовуючи оновлену модель, дозволяє виявляти шахрайські транзакції в режимі реального часу.
4. AWS Lambda використовується для виклику Amazon SageMaker, щоб зчитати дані про прийняте рішення стосовно транзакції
5. AWS Lambda виконує функцію інформування клієнтів щодо виявленої щойно шахрайської транзакції.

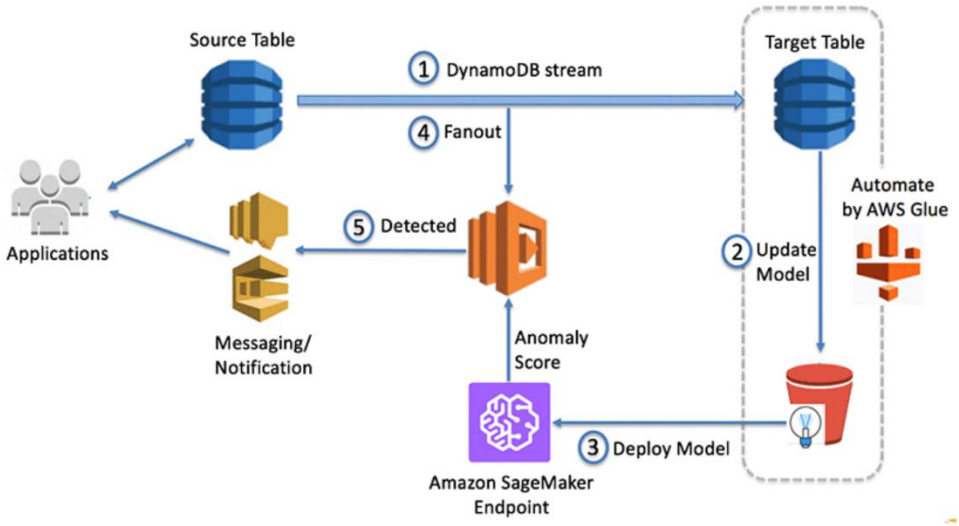


Рис. 3.18. Структура системи виявлення фроду із застосуванням AWS [25]

Масштабування системи надасть змогу використовувати її на високонавантажених проектах із сотнями операцій оплати в секунду.

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було розроблено архітектурне рішення для системи виявлення шахрайських транзакцій на основі моделі машинного начання. Із застосуванням мови програмування Python, розроблено прототип системи. Тестування і начання моделі проводилось на вибірці із 284 807 транзакцій.

Було застосовано три моделі для класифікації: Random Forest, KNN та об’єднання двох алгоритмів: логістична регресія + Random Forest. Найвища точність прогнозу досягнута при застосуванні об’єднання — 98%.

Запропоновано варіанти покращення і масштабування системи з використанням технології AWS та переходу до скорингової моделі.

ВИСНОВКИ

В даній дипломній роботі було проаналізовано сучасні та доступні на сьогодні методи визначення шахрайських транзакцій під час оплати в мережі Інтернет, або через будь-які додатки. Проведено аналіз інтеграції цих рішень в існуючий бізнес, складність налаштування та вартість цих сервісів для бізнесу.

Було розглянуто декілька найпоширеніших методів виявлення викидів у даних (пошуку підозрілих транзакцій): алгортми KNN, Random Forest, логістична регресія, Decision tree, а також застосування нормального розподілу ймовірностей.

Для вирішення поставленої задачі було протестовано алгортми KNN, Random Forest та об'єднання логістичної регресії + Random Forest. Дані алгортми було обрано, оскільки вони доволі прості в інтеграції, але при цьому мають високу точність. Модель на основі KNN добре класифікує шахрайські транзакції в обмежених наборах даних (невеликий сайт, додаток), але при нарощуванні даних рекомендовано застосовувати модель на основі Random Forest, оскільки зростає складність обчислень. Алгоритм випадкових лісів є стійким до викидів у даних і потребує менше ресурсів для начання.

Розроблені моделі були протестовані на датасеті із 284 807. Найкращі результати показав алгоритм Random Forest, метрика F1 score = 0,86, що свідчить про високу повноту вгадування шахрайських транзакцій, але при цьому і високу точність прогнозу.

Для підвищення точності роботи моделі було запропоновано варіант переходу до моделі скорингу, а не класифікації. Дане рішення дозволить встановити мануальний трешхолд для відсічення підозрілих транзакцій.

Також запропоновано варіант розширення існуючої системи із застосуванням технології AWS, що дозволить працювати при високих навантаженнях в режимі реального часу.

ЛІТЕРАТУРА

1. AFS (Advanced Fraud Detection System) [Електронний ресурс]. – AEC, 2020. – Режим доступу: https://www.aec.cz/en/afs.
2. The Only Complete Online Banking Fraud Prevention Solution [Електронний ресурс] – Режим доступу до ресурсу: https://www.threatmark.com .
3. New generation of Fraud Prevention & Risk Scoring for lending businesses [Електронний ресурс] – Режим доступу: https://www.clair.vision/.
4. Anti-Fraud Suite [Електронний ресурс]. – Режим доступу: https://www.threatmark.com/products/anti-fraud-suite-afs/.
5. Varun Chandola, Arindam Banerjee, and Vipin Kumar, "Anomaly Detection: A Survey", ACM Computing Surveys, Vol. 41(3), Article 15, July, 2009
6. В.Е.Гмурман Руководство по решению задач по теории вероятностей и математической статистике / М.: Высш. школа, 1979 / 400 с., ст. 109
7. Modern recipes for anomaly detection [Електронний ресурс] – Режим доступу до ресурсу: https://www.elementai.com/news/2019/modern-recipes-for-anomaly-detection.
8. Molnar, Christoph. "Interpretable machine learning. A Guide for Making Black Box Models Explainable", 2019
9. МЕТОДИ ВИЯВЛЕННЯ АНОМАЛЬНИХ ФІНАНСОВИХ ТРАНЗАКЦІЙ /Маслянко П.П, Миколенко О.Ю. // ПРИКЛАДНА МАТЕМАТИКА ТА КОМП'ЮТИНГ ПМК' 2019 / Маслянко П.П, Миколенко О.Ю.. – Київ: Просвіта, 2019.– С. 500.
10. Марченко О. О. Актуальні проблеми Data Mining / О. О. Марченко, Т. В. Россада. – Київ, 2017. – 150 с., ст. 41
11. Electromyographic Patterns during Golf Swing: Activation Sequence Profiling and Prediction of Shot Effectiveness [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/figure/Architecture-of-the-random-forestmodel_fig1_301638643.

12. Trevor Hastie. The Elements of Statistical Learning / Trevor Hastie, Robert Tibshirani, Jerome Friedman.. – 745 с. – (Second Edition)
13. Electromyographic Patterns during Golf Swing: Activation Sequence Profiling and Prediction of Shot Effectiveness [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/figure/Architecture-of-the-random-forestmodel_fig1_301638643.
14. Математические методы обучения по прецедентам / К. В. Воронцов // Курс Лекций Воронцова / К. В. Воронцов.. – С. 141.
15. Корреляционный анализ /М.А Харченко/ Воронеж: ВГУ, 2008. — 31 с.
16. Data Mining - lecture 6 / А.В. Гахов/Харьков ХНУ, 2014
17. Berry, Michael J. A. “Data mining techniques: for marketing, sales, and customer relationship management “/ Michael J.A. Berry, Gordon Linoff. – 2nd ed.
18. Соколов Е. А. Бэтинг, случайные леса и разложение смещения ошибки на смещение и разброс [Електронний ресурс] / Е. А. Соколов. – 2016. – Режим доступу до ресурсу: https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie
19. Jesse Davis. The Relationship Between Precision-Recall and ROC Curves
20. Мельник Б. Метрики и несбалансированные выборки [Електронний ресурс] Богдан Мельник. – 2017. – Режим доступу до ресурсу: https://ld86.github.io/ml-slides/unbalanced.html#/8
21. Соколов Е. Семинары по выбору моделей [Електронний ресурс] / Евгений Соколов. – 2015. – Режим доступу до ресурсу: https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie
22. Project Jupyter, JupyterLab Overview (2018), https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html
23. Credit Card Fraud Detection [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/mlg-ulb/creditcardfraud.

24. Resampling strategies for imbalanced datasets [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets
25. YongSeong Lee. Anomaly detection on Amazon DynamoDB Streams using the Amazon SageMaker Random Cut Forest algorithm [Електронний ресурс] / YongSeong Lee. – 2018. – Режим доступу до ресурсу: https://aws.amazon.com/ru/blogs/machine-learning/anomaly-detection-on-amazon-dynamodb-streams-using-the-amazon-sagemaker-random-cut-forest-algorithm/.

ДОДАТОК 1

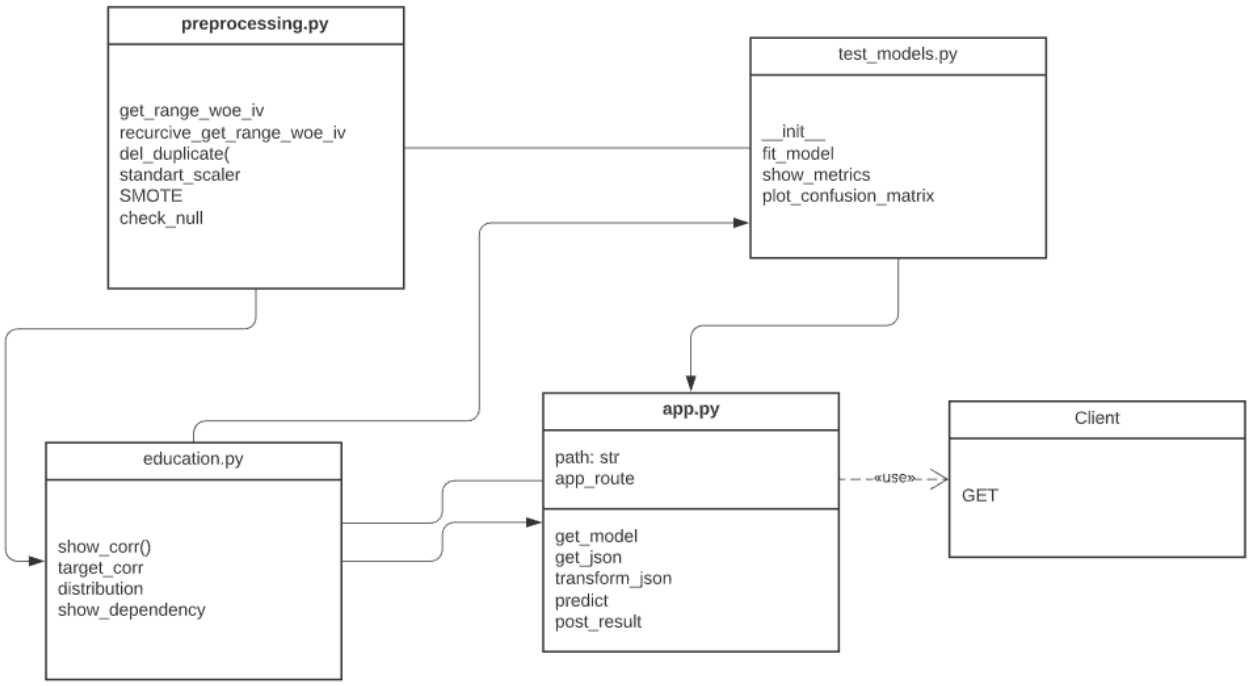
Прогнозування фродових транзакцій

Діаграма класів

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2020 р.



					ІАЛЦ.467100.004 Д1	
Зм.		№ документа	Підп.	Дата		
Розробив		Гомонець І.І.			Прогнозування фродових транзакцій Діаграма класів	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Новотарський М.А				
Н.контр.		Сімоненко В. П.				НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ гр. ІО-62
Затверд.		Стіренко С. Г.				

ДОДАТОК 2

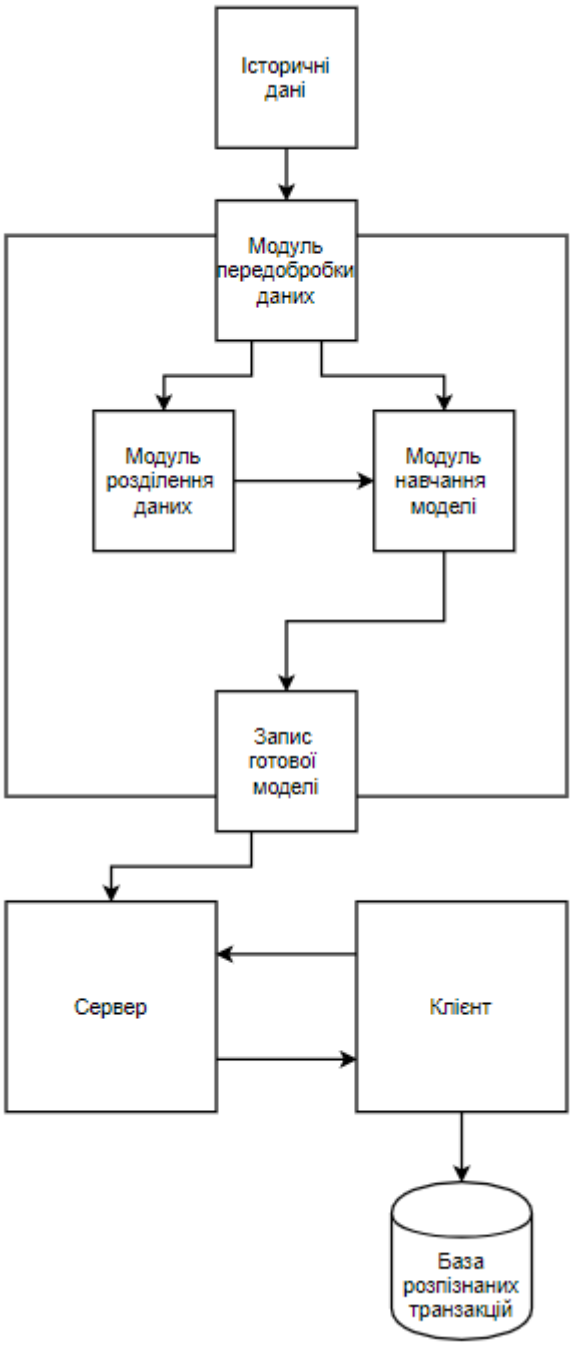
Прогнозування фродових транзакцій

Структурна схема системи

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2020р.


Історичні дані
Модуль передобробки даних
Модуль розділення даних
Модуль навчання моделі
Запис готової моделі
Сервер
Клієнт
База розпізнаних транзакцій

ДОДАТОК 3

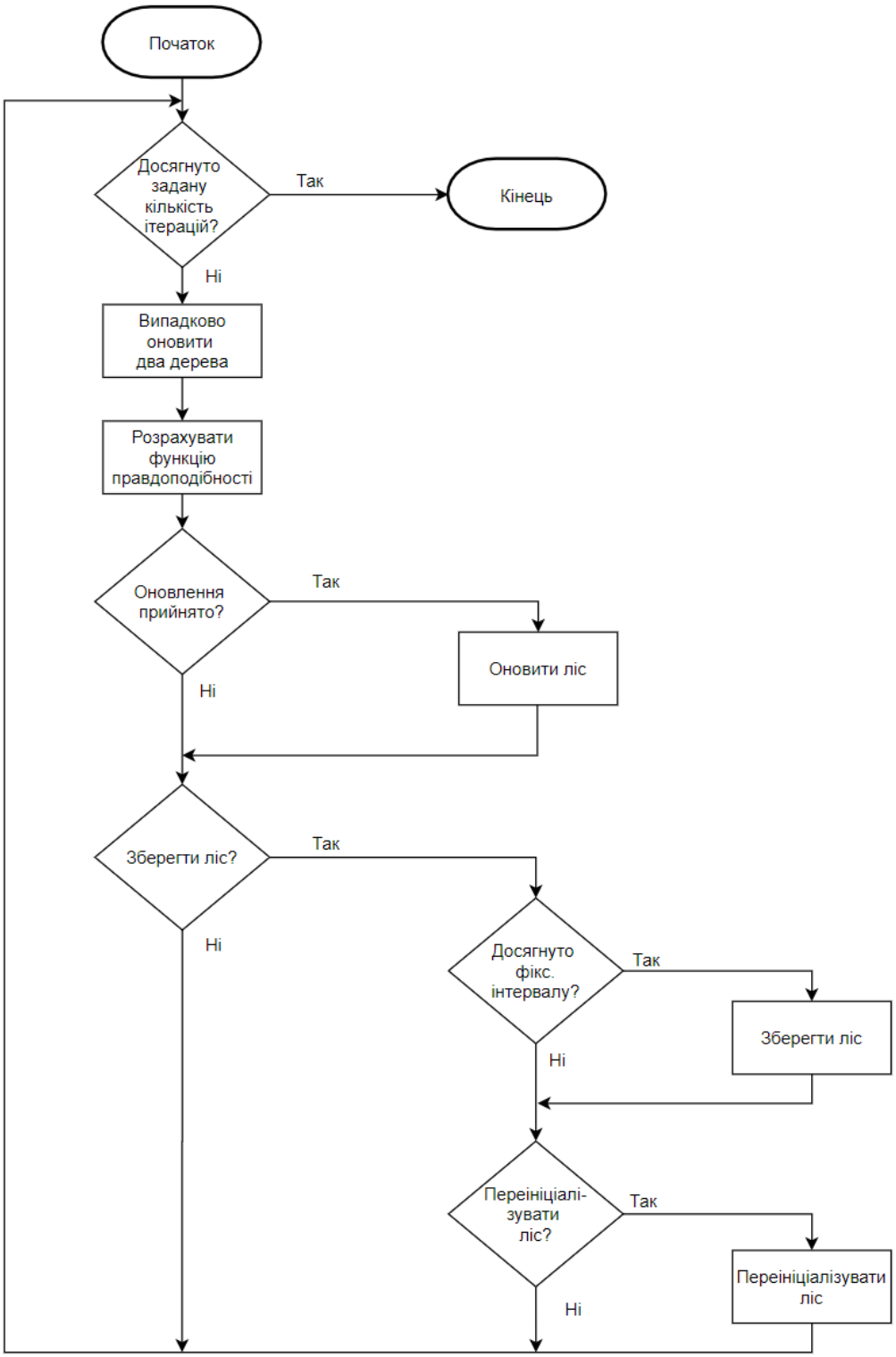
Прогнозування фродових транзакцій

Схема алгоритму моделі

ІАЛЦ.467100.006 Д3

Аркушів 1

Київ 2020



					ІАЛЦ.467100.006 Д3	
Зм.		*№ документа*	*Підп.*	*Дата*		
Розробив		*Гомонець І.І.*			*Прогнозування фродових транзакцій* *Схема алгоритму моделі*	*Літ.* / *Аркуш* 1 / *Аркушів* 1
Перевірив		*Новотарський*				
Н.контр.		*Сімоненко В. П.*				*НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ гр. ІО-62*
Затв.		*Стіренко С. Г.*				

ДОДАТОК 4

Прогнозування фродових транзакцій

Текст програми

ІАЛЦ.467100.007 Д4

Аркушів 3

Київ 2020

```
class TestModels:
    def __init__(self, model, data_train_x, data_train_y, data_test_x, data_test_y,
params=None):
        self.model = model
        self.grid_search = None
        self.params = params
        self.X_train = data_train_x
        self.y_train = data_train_y
        self.X_dev = data_test_x
        self.y_dev = data_test_y
        self.pred = None

    def fit_model(self):
        if self.params != None:
            cv = KFold(self.X_train.shape[0],n_folds=5,shuffle=True)
            self.grid_search =
GridSearchCV(self.model,self.params,scoring='roc_auc',cv=cv)
            self.grid_search.fit(self.X_train,self.y_train)
            self.pred = self.grid_search.predict(self.X_dev)
            return self.pred, self.grid_search
        else:
            self.model.fit(self.X_train,self.y_train)
            self.pred = self.model.predict(self.X_dev)
            return self.pred, self.model

    def show_metrics(self):
        if type(self.pred) != type(np.array([1,2])):
            raise Exception('Firstly fit the model!')
        else:
            if self.params != None:
                print("{:<30}{}".format('ROC_AUC score on
dev:',roc_auc_score(self.y_dev, self.grid_search.predict_proba(self.X_dev)[:,1])))
                print("{:<30}{}".format('Accuracy score on
dev:',accuracy_score(self.y_dev, self.pred)))
                print("{:<30}{}".format('F1 score on dev:',f1_score(self.y_dev,
self.pred)))
                print("{:<30}{}".format('Precision score on
dev:',precision_score(self.y_dev, self.pred)))
                print("{:<30}{}".format('Accuracy score on
train:',accuracy_score(self.y_train,

self.grid_search.predict(self.X_train))))
                print("{:<30}{}".format('Precision score on
train:',precision_score(self.y_train,

self.grid_search.predict(self.X_train))))
                print("{:<30}{}".format('Best params:',self.grid_search.best_params_))
            else:
                print("{:<30}{}".format('ROC_AUC score on
dev:',roc_auc_score(self.y_dev, self.model.predict_proba(self.X_dev)[:,1])))
                print("{:<30}{}".format('Accuracy score on
dev:',accuracy_score(self.y_dev, self.pred)))
                print("{:<30}{}".format('F1 score on dev:',f1_score(self.y_dev,
self.pred)))
                print("{:<30}{}".format('Precision score on
dev:',precision_score(self.y_dev, self.pred)))
                print("{:<30}{}".format('Accuracy score on
train:',accuracy_score(self.y_train,
```

```
self.model.predict(self.X_train))))
                print("{:<30}{}".format('Precision score on
train:',precision_score(self.y_train,

self.model.predict(self.X_train))))


                # Function is from official documentation for sklearn
    def plot_confusion_matrix(self, normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

        # Plot non-normalized confusion matrix
        plt.figure()

        if type(self.pred) != type(np.array([1,2])):
            raise Exception('Firstly fit the model!')

        else:
            cm = confusion_matrix(self.y_dev, self.pred)
            np.set_printoptions(precision=2)
            classes = ['0', '1']

            """
            This function prints and plots the confusion matrix.
            Normalization can be applied by setting `normalize=True`.
            """
            if normalize:
                cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
                print("Normalized confusion matrix")
            else:
                print('Confusion matrix, without normalization')

            print(cm)

            plt.imshow(cm, interpolation='nearest', cmap=cmap)
            plt.title(title)
            plt.colorbar()
            tick_marks = np.arange(len(classes))
            plt.xticks(tick_marks, classes, rotation=45)
            plt.yticks(tick_marks, classes)

            fmt = '.2f' if normalize else 'd'
            thresh = cm.max() / 2.
            for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
                plt.text(j, i, format(cm[i, j], fmt),
                         horizontalalignment="center",
                         color="white" if cm[i, j] > thresh else "black")

            plt.ylabel('True label')
            plt.xlabel('Predicted label')
            plt.tight_layout()
            plt.show()

app.py
import pandas as pd
from flask import Flask, jsonify, request
import pickle
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
from sklearn.base import BaseEstimator, RegressorMixin
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, learning_curve
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import roc_auc_score, roc_curve, auc as sklearn_auc,
classification_report, precision_score, recall_score, f1_score, confusion_matrix,
accuracy_score


# load model

def get_model(file_path):
    model = pickle.load(open(file_path,'rb'))
    return model

path = 'model.pkl'
get_model(path)


# app
app = Flask(__name__)

# routes
@app.route('/', methods=['POST'])

def get_jsonn():
    # get data
    try:
        data = request.get_json(force=True)
    except:
        print('Error, check json file')
    return data

def transform_json(data):
    print(data)
    # convert data into dataframe
    data.update((x, [y]) for x, y in data.items())
    print(data)
    data_df = pd.DataFrame.from_dict(data)
    print(data_df)

def predict():
    # predictions
    result = model.predict(data_df)
    return result

def post_result(result):
    # send back to browser
    output = {'results': int(result[0])}
    # return data
    return jsonify(results=output)
```

```
if __name__ == '__main__':
    app.run(port = 5000, debug=True)

preprocessing.py
target_pearson_correlations = transactions_df.drop(columns=['Time']).corr()['Class']
weak_correlation_feature_names = target_pearson_correlations[
target_pearson_correlations.abs() < 0.001
].index
display(
'Duplicated rows, Class values frequency:',
transactions_df.loc[
transactions_df.duplicated(),
'Class'
].value_counts()
)
transactions_df = transactions_df.drop_duplicates()
std_scaler = StandardScaler()
normed_amount = std_scaler.fit_transform(
transactions_df['Amount'].values.reshape(-1, 1)
)
transactions_df['Amount'] = normed_amount
display(
'Normalized "Amount" feature stats:',
transactions_df['Amount'].describe()
)
good_client_label = 0
bad_client_label = 1
def get_range_woe_iv(all_feature_values: pd.DataFrame, range_values: pd.DataFrame) ->
tuple:
    """
    Note: if in certain group there are no good / no bad clients - return WOE=0 and
IV=0
    """
    # Number of good/bad clients for the whole group
    total_target_vc = all_feature_values['Class'].value_counts()
    total_bad_clients = total_target_vc[bad_client_label]
    total_good_clients = total_target_vc[good_client_label]
    # Number of good/bad clients for the current range
    group_target_vc = range_values['Class'].value_counts()
    try:
        group_bad_clients = group_target_vc[bad_client_label]
    except KeyError:
        return 0, 0 # better to return 0 rather than np.nan or np.inf
    try:
        group_good_clients = group_target_vc[good_client_label]
    except KeyError:
        return 0, 0 # better to return 0 rather than np.nan or np.inf
    # Calculate "event pct" / "non event pct"
    good_clients_distr = group_good_clients / total_good_clients
    bad_clients_distr = group_bad_clients / total_bad_clients
    # Calculate WOE and IV
    woe = np.log(good_clients_distr / bad_clients_distr)
    iv = (good_clients_distr - bad_clients_distr) * woe
    return woe, iv
```